Integrated E-Assessment

Habilitationsschrift

im Fach Informatik

vorgelegt an der

Fakultät für Wirtschaftswissenschaften

der Universität Duisburg-Essen
Campus Essen

vorgelegt von
Dr. Michael Striewe
im Mai 2022

Datum der mündl. Habilitationsleistung: 30. Januar 2024

Abstract: Assessments are an important part of many teaching and learning activities. Since several decades, electronic aids are involved in planning, conducting and evaluating such assessments. However, there are still many scenarios in which electronic assessments cannot be smoothly integrated into teaching and learning due to organizational or technical issues. Although there is a large body of research on the educational aspects of electronic assessments, there seems to be a lack of research on the connections between organizational and technical aspects as well as on the integration of technical innovation into a larger context. This thesis aims to reduce that lack and to create a map that defines, explains and covers electronic assessment as a subject of study that brings together research results from process modeling, software engineering, educational technology and educational data analysis. This thesis summarizes and advances the state-of-the-art in these four areas by creating structured catalogues and conceptual frameworks. In addition, it maps these theoretical results back to practice in several case studies. The result is an integrated view on various aspects of electronic assessments and their connections between each other and a larger context, which is intended to prepare the ground for future research.

Acknowledgements: There are many people who deserve my warmest thanks and gratefulness for their support through all the years that I worked on this publication: Michael Goedicke for leading a great research group that allowed me to evolve my skills, pursue my scientific interests and shape my academic profile; all my colleagues in that group for many discussion on almost each and every aspect of e-assessment, as well as for picking up some of my duties when I needed time for research and writing; Marco Konersmann for being a great office mate and for sharing lots of brilliant groaners; Sven Strickroth for many thoughts on educational technology in general and e-assessment in computer science in particular, as well as for co-organizing several workshops; Eric Ras for valuable conversations on automatic item generation; Peter Hubwieser, Marc Berges, Johannes Krugel and Mike Talbot for their collaboration in the field of computer science education and competency measurement; Justin Timm und Thilo Schramm for their collaboration on e-assessment in biology; Carolin Eitemüller and Florian Trauten for their collaboration on e-assessment in chemistry; Till Massing and Christoph Hanck for their collaboration on learning analytics; Meike Ullrich and all colleagues at KIT, DFKI and UP for launching the KEA-Mod research project; all students who supported my research with their work in seminars, projects and theses; all organizers of the German "E-Prüfungs-Symposium" who allowed me to run several workshops and to deliver a keynote talk; all anonymous reviewers who accepted my papers, but even more all anonymous reviewers who rejected my papers with valuable advice on how to improve them; and finally my family, my wife and my son for every encouraging and supportive smile.

Contents

List of Figures

List of Tables

1. Introduction

Modern education makes use of electronic aids in many different forms. Educational technology can help in many places in the teaching and learning process. Some technology-enhanced teaching or learning systems may only help in very specific situations, while others may be of broader interest. Technology-enhanced assessment systems (called "e-assessment systems" for short) can be considered to be one of the latter kind, since assessments appear in many places throughout teaching and learning activities: Diagnostic assessments can help to plan learning activities, formative assessments can steer learning activities, and summative assessments can check the learning outcome. One can thus assume that e-assessment plays a central role in modern education and deserves some deeper consideration in research to ensure a high quality of solutions.

Indeed, e-assessment is a subject of research for more than 60 years in some domains [124]. Recent literature has confirmed that there seems to be no major difference between assessments with and without electronic aids [294] and that students have a positive attitude towards e-assessment [68]. However, the oldest publications on e-assessment report about e-assessment systems using punchcards, which is obviously not what we use today. Hence, we should have a look on how the current state-of-the-art in software system design can help to design e-assessment systems and connect them to other systems that are relevant in the context of teaching and learning. Similarly, some publications report on organizational issues that arise from the introduction of electronic tools into assessments and thus we should also have a look at the organizational processes of assessments to find out how electronic aids are integrated in that area. Other aspects may come up with similar questions and we must also consider that there may be differences between different domains of study. For example, we can speculate that e-assessments could be integrated very well into teaching and learning of algebra, but less integrated into teaching and learning of archaeology. These examples also point out that the different aspects cannot be viewed solely in isolation. Different domains may have different traditions in the way they conduct assessments and thus also have different processes and thus different organizational issues. In turn, they may also require different technical solutions. Thus viewing each of these aspects in isolation cannot advance the field of e-assessment in the same way as an integrated view can do.

Recent research also notes "that much research and innovation happens in silos, where policy, research and practice on assessment, technology enhanced assessment and ethical and political concerns are not linked up" [294]. However, it is quite obvious that a close integration between research on educational technology on the one hand and particularly software engineering research on the other hand is necessary to create actually usable systems. Algorithms and methods that realize high quality educational support will not help, unless they are integrated in systems that are dependable, scalable, secure and

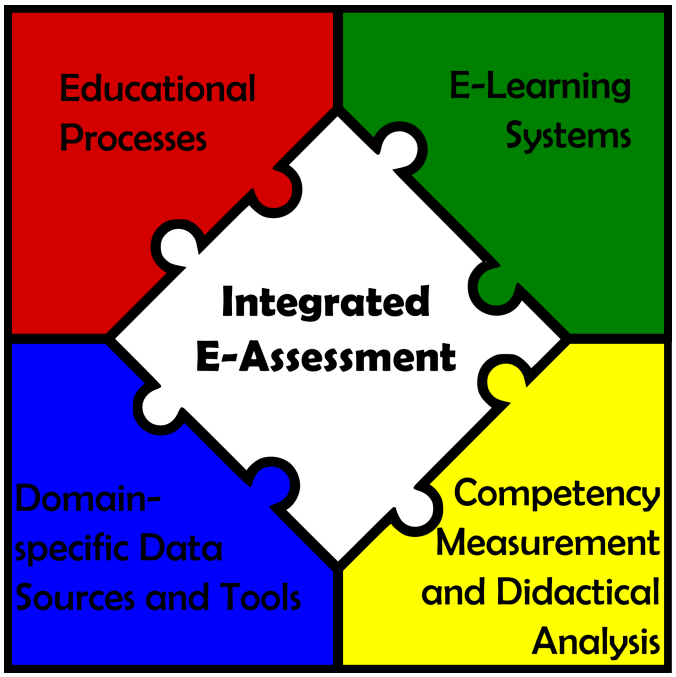


Figure 1.1.: Integrated E-Assessment and its Context

extensible. At the same time, systems with these quality characteristics are only truly helpful in breadth and depth of education if the allow to integrate very specific needs of some domain of study or educational scenario.

Figure 1.1 shows four different areas in the context of e-assessment and all of them are relevant to the question of integration. Any of these areas has its own experts, its own core of research and its own state-of-the-art. An e-assessment system can contribute to each of these areas and can also reuse results existing in any of these areas. Thus it seems to be beneficial to connect research from all these areas and to define a joined view on e-assessment, which in turn can hence be integrated seamlessly into all these areas. This is what this thesis aims to do. The idea is to create a kind of a map or picture of connections between various aspects of e-assessment, to see all the connections that actual make up the integration. In that sense, this thesis does neither ask a single question nor try to find a final answer. Instead, it is guided by an array of questions and tries to define a starting point for future research efforts.

1.1. Scientific Questions and Contributions

The goal of this thesis is to create scientific connections between studies on the general and domain-specific use of e-assessments as a means of higher education on the one hand and computer scientists working on e-assessment systems and features as a subject of software development and engineering on the other hand. The aim is to define, explain and cover e-assessment as a subject of study that brings together research results from

process modeling, software engineering, test item design and educational data analysis. Only joining these views allows to integrate e-assessment seamlessly into the every-day duties of teachers and assessors to make it usable exactly in the way and to the extent it is needed.

From the scientific point of process design and modeling, this thesis asks the question on what change to traditional assessment processes is necessary if e-assessment options are integrated. Before this can be answered, this thesis first defines a notation for expressing assessment processes based on extensive literature research. This helps to unify the terminology from current scientific discussions, and to find a way to structure and compare scientific contributions to different aspects of assessments.

Software engineering mostly contributes to the development of e-assessment systems based on general findings on aspects like performance, security or usability, but without considering educational technology as a dedicated domain of software engineering. This thesis answers questions about the specific characteristics of e-assessment systems as one kind of educational technology. In particular, it creates catalogues of typical designs of e-assessment system or features. This helps to sharpen the understanding of essential qualities e-assessment systems must ensure by design and thus to identify aspects that need further research. Moreover, it allows to compare design decisions within educational technology with other domains in which similar catalogues are available.

Independent of processes and systems, test items are still the heart of educational assessment in any of its forms. So even if e-assessment is integrated seamlessly into existing processes and systems, it may not be applicable in domains in which no appropriate test items exist to represent the important contents and competences of that domain. Hence this thesis also looks at the available techniques to create test items that represent the contents and competences of these domains as accurate and complete as possible. At the same time, the question is asked how domain-specific data sources and tools can be used for automated generation or evaluation of test items. This helps to ensure that e-assessment is not decoupled from data and tools actually used in a particular domain and contributes to an integration of research and teaching in the respective domains.

Finally, modern competence-oriented assessments are based on psychometric analysis of data which is collected during the assessments. Many other applications like learning analytics rely on data from e-assessment system as well. Thus it is of interest which data can be generated and provided by e-assessment systems in order to ensure and improve assessment and education quality. This thesis thus investigates how statistical methods can be used to interpret assessment data and how that is embedded into the context of actual assessments. This helps to understand e-assessment as an enabler for evidence-based improvement of education and raises some remarkable legal and ethical issues about personal data at the same time.

1.2. Structure of this Thesis and Previous Work

This thesis will discuss the topic of integrated e-assessment from four different points of view along the following path:

- First, part I takes an organizational perspective that discusses the abstract process of conducting educational assessments and the possibilities and dependencies for integrating electronic tools into that process. It is to a large extent based on a previously published book chapter [272] and a conference paper [276], but provides much more details.

- Second, part II takes a technical perspective that discusses software engineering questions with respect to the integration of e-assessment tools into general e-learning-systems or learning-management-systems. Preliminary results of that research have been published earlier on workshops and conferences [271, 274, 275]. One of the case studies has also been used in a slightly different context [269], while research results from that part have also influenced the development of another e-assessment system [284].

- Third, part III takes the perspectives of different domains to discuss the various ways of formulating domain-specific problems in a suitable way for e-assessment. A particular focus of that part is on the automated generation or evaluation of item types based on domain-specific data sources or tools. It includes results from various joint and independent research projects that have published individually as journal, conference or workshop papers (i. e. [248, 154, 282, 270, 273, 216, 217, 293, 283] amongst others).

- Fourth, part IV takes the perspective of data analysis to discuss how e-assessment can help to elicit, identify or analyse data which is necessary for statistical and psychometric approaches in education. That part includes a literature review that has been published as a conference paper [277]. It also partially builds upon experiences from joint and independent research projects that have published their results earlier as journal, conference or workshop papers (i. e. [126, 186, 150] amongst others).

Each of the four parts applies a general research method: In a first step, existing literature is gathered and structured to create an overview on the state-of-the-art or to create catalogues and classifications of existing approaches. In a second step, selected aspects are studied in depth to fill missing gaps, advance the field and create meaningful, directly applicable contributions to e-assessment systems. As a final step, each of the four parts is concluded by a summary of results with respect to integrated e-assessment as well as with respect to a wider scope beyond the focus of this thesis.

The final part V of this thesis concludes the results seen so far to create a final big picture of integrated e-assessment. It also reviews the achievements of this thesis as well as the future research that can be started from these results.

Part I.

The Educational Assessment Process

2. Educational Assessment as a Matter of Organization

Preparing and conducting educational assessments is not an easy thing. First of all, the contents have to be created carefully to make sure that the assessment considers right what is right and wrong what is wrong. Second, test pedagogy and psychometry have to be considered to make sure that the assessment really measures what it is supposed to measure. Finally, a lot of organizational aspects concerning the when and where have to be considered even for informal low-tech assessments, where neither assessment authorities nor electronic assessment tools are involved.

The first two aspects are at least to a larger extent covered in the domain specific or general education of educators, respectively. Similar to the organizational issues, they are relevant for any kind of assessment. Moreover, they are not supposed to get significantly worse when using electronic assessments instead of non-electronic ones. Teachers might need to take limitations of electronic devices into account, but as traditional ways of conducting written or oral exams have their limitations, too, this is no specific obstacle. In contrast to that, organizational issues indeed tend to get worse when using electronic systems. Although using electronic systems may help to reduce effort or automate steps, it also adds other steps to do or people to involve in the process of assessment. Thus one aspect of integrated e-assessment is to integrate the use of electronic systems into the organizational process of assessment. This is the aspect discussed in this part of this publication.

To do so, it first defines a way of modelling assessment processes that can be used both for electronic and non-electronic assessments (chapter 3). Then several cases are discussed, where some of them use electronic systems and some not (chapter 4). This leads to the desired analysis on what change to traditional assessment processes is necessary if e-assessment options are integrated (chapter 5).

Formal process models for educational assessment cannot only be used to define the differences between electronic and non-electronic assessments. Similar to process models in other domains, they can be used as a means of quality and risk management or as a means of comparison and assessment. The former does not only involve general technical or organizational risks like failing assessment systems or late delivery of printed exam sheets, but relates also to the legal aspects of educational assessments. Formalized processes can help to document the proper and law-abiding conduction of assessments, which is relevant at least in the context of some summative assessments like final exams.

With respect to comparison and assessment, educators may be faced with decisions on which kind of assessment they construct and conduct (e. g., whether they use a written or oral exam) or which technical assessment tools or systems they use. In order to make

an informed decision between different options, educators need to compare different attributes such as efficiency or reliability. Formal process descriptions can help in that case, too, as they can be used to estimate costs (not necessarily in terms of money, but in terms of activities to be performed) or identify the number of steps that bear the risk of having negative influence on the reliability.

Consequently, the idea of defining an universal way of modelling assessment processes is not only relevant in the context of integrated e-assessment, but also creates a benefit for the general handling of educational assessment. Hence this part of this publication also contributes to a much wider scientific discussion beyond the scope of this publication.

2.1. Literature Study

Before we can start to design and describe a general process of educational assessment, some things have to be verified or checked by literature study. First, some evidence is needed that educational assessment is indeed seen as a process by researchers and educators. Second, existing process descriptions have to be analyzed and compared to find out whether they qualify as components for a general process description. Third, a suitable notation has to be identified that is both easy enough to use and powerful enough to allow to express and compare all relevant properties.

2.1.1. Process Properties of Educational Assessments

In general, there is no doubt in recent literature that educational assessment is a process [227, 135]. Typical terms related to processes in general such as “phases”, “cycles”, “time line” or “matrix of activities” can also be found in literature on educational assessment [24]. They are even more common for the more general area of Instructional System Design [74, 226]. Hence it can be assumed that it is in general suitable to describe educational assessment by means of process modeling.

However, it has to be recognized that the notion of a model or process as found in the literature in conjunction with educational assessment is not necessarily used in the meaning of a concrete and formal process model. It can also refer to more abstract and informal models such as the one defined in [59]. Literature studies such as the one presented in [304] analyse different assessment process models and identify common features. These models and analyses demonstrate that in educational assessment some activities happen repeatedly following some definition, although it is only weakly formalized. Some authors also use the term “process” as a synonym for an activity that takes both time and preparation without referring explicitly to the organizational aspects of an assessment. Examples for this are definitions like “Assessment is a multidimensional process of judging the individual in action” [166] or “assessment is the process of defining, selecting, designing, collecting, analyzing, interpreting, and using information to increase students’ learning and development” [83]. These definitions that do not refer to organizational aspects in any way are not considered in this literature study.

It is also common in the domain of education to understand training and instruction as a system which follows not only a process definition, but also has input and output

[158]. If input is divided into “people”, “material”, “technology” and “time”, this can also be applied specifically to educational assessment: *People* consist of at least organizers and participants of the assessment, *materials* are the functional contents which reflect the scientific or professional domain of the assessment, and *technology* may refer to the physical way of delivering the assessment.

2.1.2. Specific Aspects of Educational Assessments

Besides the general notion of a process, literature on educational assessment frequently also reports on specific challenges in the course of preparing and conducting an assessment, that are closely related to processes. In each of the cases discussed below, process models can be used as formal means to support in tackling these challenges successfully. Notably, none of these cases is special to educational assessment in the sense that it does only occur in that context. Instead, these cases illustrate challenges that are commonly tackled with support of process models also in other domains, but that are particularly interesting in the context of educational assessment.

Team Management and Communication

In many cases, more than one person is involved in organizing an educational assessment [227, 24, 135] and a single person may act in many different roles [249]. Consequently, persons have to communicate and collaborate in order to prepare and conduct an assessment and also have to be aware of the different responsibilities they have due to their respective roles. Case studies such as [173] list several issues that may occur when using a new way of assessment in practice and a large amount of these issues is concerned with staff communication. Also explicit advice to create a written plan for an assessment to be able to track progress can be found in literature [24, 13]. Hence we can assume that a proper process model can be used in this domain to support communication, assignment of tasks, and process monitoring.

Quality Improvement

As educational assessments are conducted quite often and at least some assessments may have fundamental impact on the future of the assessment candidates, quality goals like efficiency or high reliability and validity of exams exist [255]. In this case, formal process descriptions are used as a means of quality improvement as they allow to document necessary steps and thus help to define and keep a successful way of working.

Risk Management

As educational assessments deal with personal data and might be subject to malicious attempts, risk management techniques can be applied to handle these issues. As part of an analysis on the state-of-the-art in risk management an information reference model according to ISO/IEC 23988 has successfully been created for computer-assisted assessments [50]. Another formal risk analysis has been published earlier, identifying just

risks but no process steps and focusing more on the transition from classical assessments to computer-assisted assessments [120]. Both publications provide evidence that educational assessments or at least computer-assisted assessments have several similar properties as classical business processes with respect to risks and the suitability of using risk management techniques.

2.1.3. Existing Formal Process Descriptions

As mentioned at the beginning of this section, not only informal but also formal process descriptions can be found in literature. The following subsections exemplarily discuss models for local situations as well as generic process models. The purpose of this review is to create a list of process elements that are common to both kinds of models. This list will be summarized towards the end of this section and be used as an input for chapter 3.

Process Models for Local Situations

An attempt to model the process of conducting paper-based exams in detail can be found in [145], which is based on a master thesis [310]. The process model is used as a means of requirements engineering in the context of software support for large paper-based exams. A graphical modelling language for detailed modeling of business processes is used. Besides course-grained actions like "Klausur erstellen" (eng.: Author exam) or "Korrektur der Bewertung" (eng.: Grade exam) also fine-grained actions such as "Klausuren und Deckblätter austeilen" (eng.: Hand-out exams and cover sheets) or "Daten auf Deckblatt schreiben" (eng.: Write data on cover sheet) are listed in the model. The model also makes a distinction between roles by naming e. g. "Prüfungsamt" (eng.: Exam Authorities) and "Studenten" (eng.: Students). Finally, the model divides the process into five phases for the creation of the exam, preparation of the assessment, conduction of the assessment, post-processing (i. e. evaluation), and review and archiving. Although the study only aims to model the situation in a particular department, we can use it in two ways: First, we can pick roles, activities and phases used there and compare to other models to find out what is general and what is specific. Second, we can use the model as evidence that process models for educational processes are considered helpful in the context of software design for supporting systems and that using distinct roles is considered useful in this context.

The same is true on a slightly more abstract level for the process models presented in [62], which are related to a university-wide process, but still limited to a particular existing tool. Again we can identify roles like "Exam Office" or "Students" in conjunction with their activities. In addition, we can use the publication as an evidence, that model are not only used in requirements engineering for new tools, but also for documenting the use of existing assessment tools.

General or Generic Process Models

As part of the FREMA (Framework Reference Model for Assessment) project a domain definition for assessment in the context of e-learning was created. One part of this domain

definition is a concept map for e-learning assessment processes [190, 309]. This map is based on interviews within the assessment community and lists activities that turned out to be relevant based on these interviews. It covers several didactic aspects such as authoring of assessment items, checking solutions for plagiarism or creating feedback to students, but also organizational issues such as checking the availability of candidates and staff or preparing digital and physical environments. In a second step, the domain model has been mapped against existing software to find out which are covered to what extent by educational software. Independent of these mapping results, which are hardly usable more than 10 years later, the concept map for e-learning assessment processes is another valuable evidence that explicit process modelling is considered useful and that practitioners are aware of a lot of different didactic and organizational activities related to assessment.

An approach to create generalized process descriptions for educational processes by reverse engineering from e-learning tools is presented in [117]. The result of the approach are activity-oriented workflows that are created by mapping tool-specific activities and actors to generic descriptions. The purpose of the models is to define educational cloud services in which actors such as teachers can do their work in terms of process steps independent of the implementation of these steps in an actual e-learning system. A similar goal is pursued in [169], but the approach there is to define business flow diagrams and data flow diagrams in a top-down manner and to derive key system modules from these models later on. In both cases, the resulting models are less fine-grained than the one discussed above, but also make a distinction between actors such as "Teacher", "Learner" or "E-Assessment System". Hence we have some more evidence that process models for educational processes are considered helpful for designing educational software services and that distinct roles are considered useful in this context. The same conclusion is supported by the process analysis presented in [170].

The IMS Question and Test Interoperability specification also provides a partially formal process description of their understanding of assessment processes during the introduction of their standard [131]. The standard does not define process steps of any kind, but actors and use cases relevant to the standard.

Another example is the process model for the preparation and processing of programming assignments as presented in [296]. In fact, the model is not specific to programming assignments, but summarizes the main steps starting from the composition of an assignment up to the review of the marked assignment by the student on a course-grained level.

Other Formal Models for Assessment

There is also some amount of research which is not concerned with modeling the whole assessment process on a rather abstract level, but modeling parts of it in detail. An example for this is the graph-based modeling of admission criteria for exams, which is based on activities, categories and rules [264]. It allows to formalize fulfillment of admission criteria from an operational point of view. It is thus on the one hand a formal model for a particular detail of the assessment process (i. e. the activity to find out who

Actor Name and Synonyms	Short Description	References in Literature
Student (also: Candidate, Learner, Test-taker)	A person who is supposed to sit the exam and to answer questions in the test.	[135] [24] [255] [249] [145] [309] [117] [50] [227] [115] [156] [170] [138] [212] [131] [296] [193] [169]
Teacher (also: Author, Examiner, Faculty, Instructor, Professor)	A person who prepares and conducts assessments and decides about grades and feedback. May also be the one who creates assessment items and designs tests.	[135] [24] [255] [249] [145] [309] [117] [227] [115] [170] [138] [212] [131] [296] [193] [169]
Exam Authorities (also: Exam Office, Departmental Secretary)	An institution responsible for formal or organizational aspects of assessments.	[255] [249] [145] [309]

Table 2.1.: Different actors found in formal assessment process descriptions in literature.

may take part in an assessment) and on the other hand a model which is able to connect different assessments with each other (i. e. to relate assessment outcomes) independent of their individual processes.

Summary of Process Elements

From the different existing formal process descriptions a list of concrete elements and element types can be compiled which occur in at least some of the descriptions. These elements and types are candidates for elements to be included in a generalized and universal process model for educational assessment that will be developed in chapter 3.

Table 2.1 lists three different actors that commonly appear in literature. They are listed with synonyms for their names that can be found in various publications. Some sources in literature make stronger differences between more actors which are aligned here to a more general set. The most important distinction is the one between people developing tests and people using tests, that can be found for example in [227] and [131]. The idea is that domain experts create assessment items and compose meaningful tests from them, while teachers may use these tests to assess their students. Although there are surely many assessments conducted this way, there are also many in which teachers themselves author assessment items or at least amend items they picked from an item pool. Moreover, there are also assessments in which teachers use items or complete assessments they authored years ago. In these cases it is nearly impossible to draw a

Concept Name and Synonyms	Short Description	References in Literature
Question (also: Assessment Item, Test Item, Assignment)	A single item within an exam which can be answered by a student independently of other items.	[135] [24] [249] [309] [50] [227] [170] [138] [131] [296] [193] [169]
Exam (also: Test, Question Set, Assessment, Quiz, Test paper)	A collection of questions that is delivered to the students.	[135] [24] [255] [249] [145] [309] [117] [227] [156] [170] [138] [131] [193] [169]
E-Assessment System (also: Digital Environment, Tool, Exam Server)	An electronic system used in the assessment process mainly for delivering exams, collecting responses or creating feedback.	[13] [309] [50] [138] [212] [131] [193] (also in [117] and [156] as actor)
Room (also: Physical Environment)	The location where students are supposed to be while sitting the exam.	[255] [309] [310] [170] [138]
Feedback (also: Grade, Score, Results)	The pieces of information produced to describe and inform about the exam results.	[135] [24] [255] [309] [310] [117] [50] [138] [212] [131] [296] [193]

Table 2.2.: Different objects or concepts found in formal assessment process descriptions in literature.

sharp border between people developing tests and people using tests. However, one could define both roles but having only one actor in some processes filling out both of them. In this case one could argue that teachers frequently include pictures or diagrams created by someone else into an assessment, but that would not make the original artist or illustrator an actor in the assessment process. Following this argumentation a person just creating questions and tasks but not directly using them for assessment is also not an actor in the actual assessment process. Consequently, it seems to be sufficient to have one actor who prepares and conducts the assessment and who may or may not be the author of the test items as well.

Other actors than the ones listed in the table appear also in the literature. However, they seem to be passive and only acting on demand of one of the actors named above. This includes staff like proctors or invigilators monitoring assessments, tutors helping students to review their results or technical staff helping to set up the assessment environment.

Table 2.2 lists five different concepts or objects that commonly take part in formalized assessment processes in literature. They are listed with synonyms for their names that can be found in various publications. There are more concepts that can be found in literature, but they do not appear to be common for assessment processes. This applies in particular to concepts related to physical objects such as exam sheets, which do neither appear in electronic assessments nor in oral assessments.

Table 2.3 lists activities that appear in formalized assessment processes in the literature. They are listed with synonyms for their names that can be found in various publications. A lot more activities can be found in the literature, but not all of them appear to be universal. This applies in particular to activities that are related to concepts or objects that are not of universal relevance, such as copying question sets for preparing exam sheets (as found in [255] and [145]) or maintaining e-assessment software (found in [249]). The latter is also a bit out of scope for this summary, as software maintenance is an ongoing duty of an IT-department and not an activity specifically related to assessment processes.

Activity Name and Synonyms	Short Description	References in Literature
Describe goal of assessment	Document the intended scope, content and outcomes of the assessment.	[24] [13]
Prepare grading schemas	Create rules for grading answers and deciding on pass and fail.	[145] [309]
Author items (also: compose assignment)	Create assessment items and related materials.	[135] [24] [249] [309] [227] [170] [138] [296] [193] [169]
Author tests	Compose tests by combining assessment items.	[255] [249] [145] [309] [227] [170] [138] [193]
Assure test quality (also: Control and sign)	Verify tests to conform to the goals of the assessment.	[255] [249] [309]
Schedule	Define where and when the assessment will be conducted.	[145] [309] [170] [138]
Register students	Create a list of candidates allowed and supposed to take part in the assessment.	[145] [309] [156]

Table 2.3.: Different activities found in formal assessment process descriptions in literature (continued on next page).

(Table continued from previous page)

Activity Name and Synonyms	Short Description	References in Literature
Allocate staff	Make sure that teachers, administrators, assessors, invigilators or similar join the assessment as needed.	[309] [310]
Prepare environment	Set-up physical or electronic devices used for the assessment.	[309] [193]
Authenticate students	Ensure only authorized candidates take part in the assessment.	[145] [309] [50] [156] [170] [138] [193]
Distribute and collect exams (also: Transport, Hand-Out, Deliver)	Make tests available to candidates and collect their answers.	[255] [145] [309] [117] [50] [156] [138] [296] [193]
Sitting tests (also: Writing to exam sheets, Answering tests)	Respond to assessment items by creating answers.	[255] [249] [310] [117] [227] [170] [138] [296] [193] [169]
Monitor exams	Observe candidates taking part in the assessment to detect malicious attempts.	[145] [170]
Check answers for plagiarism	Analyse answers given be candidates to detect malicious attempts.	[309]
Create feedback (also: Grade answers, Mark)	Analyse answers given be candidates to judge and comment them.	[135] [24] [255] [249] [145] [309] [117] [50] [170] [138] [296] [169]
Report grades	Provide feedback on answers to candidates.	[255] [145] [309] [117] [50] [138] [296] [193]
Handle grades review (also: Moderate, Appeals)	Reply to complaints about potentially misjudged answers.	[145] [309] [117] [50]
Analyse assessment results (also: Access statistics)	Check for conformance of the assessment results with the goals of assessment.	[24] [13] [249] [309] [117] [50] [170] [193] [169]

Table 2.3.: Different activities found in formal assessment process descriptions in literature (continued on next page).

(Table continued from previous page)

Activity Name and Synonyms	Short Description	References in Literature
Improve assessment	Draw conclusions from results to be used in the next assessment.	[24]

Table 2.3.: Different activities found in formal assessment process descriptions in literature.

2.1.4. Available Process Notations

The purpose of this section is to identify a formal process notation which is suitable for expressing educational assessment processes. The notation should be usable in a descriptive manner as well as in a prescriptive manner. In particular, it should be possible to derive concrete tasks for conducting an assessment from the process description and also to monitor the state of an ongoing assessment process. Moreover, process descriptions should be customizable with respect to the actors or concepts included in the process, so that it is possible to describe e-assessments as well as assessment involving no e-assessment systems. Finally, it should be able to support agile and informal assessments in which no specific time constraints exist or in which some activities can happen in arbitrary order.

There is a wide range of notations for processes focusing on different aspects and being either general or domain specific. One of the earliest and still popular graphical notations are Gantt charts [94, 51], that organize activities in a timeline. More recent notations like process flowcharts [247], event-driven process chains [300] and similar focus on dependencies by showing activities as nodes in a graph. Some also add decision and processing nodes to the graph. There exist diverse variants of flowcharts, for instance, used for decision making in medical applications [303] or tailored to business process visualization [230]. However, all these notations focus on strict and precise scheduling of tasks or strict sequences of activities in a highly automated or constraint context. Hence they do not meet the requirements for a process notation for educational assessment processes with respect to support of agile assessments. Support for these can be found in Kanban boards known from software engineering [128], but these in turn focus solely on monitoring and informal planning of tasks. Thus they do not support a structured and prescriptive documentation of processes.

The ESSENCE standard [2] origins also from the domain of software engineering and extends some of the ideas from Kanban boards to tackle process related aspects of software engineering projects. It defines both a notation for software engineering process descriptions as well as a so-called kernel of key elements (named “alphas” and “activity spaces”) that are supposed to be relevant in any software engineering project. Each alpha defines a set of states with checklists, which allow to track project progress. An alpha state is considered achieved when all items on its checklist are done and all preceding states of the alpha have also been achieved. Simple process descriptions can be created by forming groups of states from several alphas and thus defining project phases or milestones. More

details can be added to a process description by assigning definitions of documents called "work products" to alpha states (e. g., to represent the alpha "Requirements" by a work product named "Use Case Documentation") or definitions of tasks called "activities" to activity spaces. Hence a process description using the ESSENCE notation can be used prescriptively as well as descriptively. In a descriptive way, checklist items of states can be ticked off to determine the current state of the process and activities that are already completed. In a prescriptive way, activities to be performed next can be derived from checklist items that are not yet checked. At the same time, the notation still supports agile enactment of processes as it makes no restriction on the order in which checklist items are ticked off.

Although the ESSENCE standard defines both kernel and notation, these parts do not strictly depend on each other. While its kernel contains elements relevant in the context of software development processes, its general notation and notions are usable for a much wider range of processes. Hence it is possible to use the ESSENCE notation also in other domains by defining a different kernel. As the ESSENCE standard expresses the state of concepts and objects (such as "Requirements" or "Software System") as well as the state of actors (such as "Stakeholders" or "Team") as alphas, it is possible to map the contents of table 2.1 and table 2.2 to alphas. Contents of table 2.3 can then be mapped to activities. Thus it seems to be possible to define a kernel for educational assessment using ESSENCE.

With respect to the integration of electronic systems into assessment processes this is also an interesting choice because of the concept of "work products" mentioned above. Considering a written assessment, the same exam (as a conceptual entity) can be represented by an exam sheet (as physical representation) or via the user interface of an e-assessment system (as digital representation). In ESSENCE this is reflected by the fact that a kernel may contain an alpha "Exam" representing the conceptual entity, while detailed information on how to represent this entity are added later.

It is interesting to note that ESSENCE was already chosen as a means of modeling educational practices in other contexts. In [287] an extension to the standardized ESSENCE kernel is presented in which a learning environment and related learning and teaching activities are considered for teaching embedded systems design.

3. The Essence of Educational Assessment Processes

The goal of this chapter is to elicit a list of core concepts and terms that are required to describe the process of educational assessment. It is based on the findings from literature that are summarized in section 2.1.3. These concepts and terms are presented in form of a so-called "kernel". This way of presentation follows the ESSENCE standard which does the same for software engineering as discussed in section 2.1.4.

3.1. A Kernel for Educational Assessment

The ESSENCE Kernel for Educational Assessment is intended to form a common base for all kinds of educational assessment processes. It is neither limited to a particular didactic purpose of the assessment (e. g. diagnostic, formative or summative) nor to a particular form of assessment (e. g. written assessment, oral assessment or electronic assessment). However, a particular focus of the following sections will be to demonstrate how the specific needs of e-assessments integrate into the general process of educational assessment.

In order to achieve full but flexible coverage of all kinds of processes in educational assessment, the Kernel consists of eight Alphas from which two are optional. The kernel also contains Activity Spaces, which are discussed later in section 3.1.3. Alphas and Activity Spaces can be grouped into three Areas of Concern somewhat similar to the original ESSENCE Standard. Notably, the decision to use an electronic system in the assessment just adds a particular alpha to the process, but does not necessarily change the whole process. An overview of the kernel alphas and their main relationships is shown in figure 3.1. The relationships shown in this figure are those that are used throughout the following sections when explaining the meaning of the different alphas in more detail. Nevertheless, additional relationships may possibly exist, but are not discussed here.

Following the original ESSENCE specification, the kernel does neither include Work Products associated with the Alphas nor Activities associated with Activity Spaces. These can be added later to cover practices on how to conduct assessments in a particular domain or in a specific manner. They are out of the scope for a universal kernel and hence only discussed by example in the case studies in chapter 4.

3.1.1. The Areas of Concern

As discussed in section 2.1.1, one can divide the input to a process into the categories "people", "material", "technology" and "time". We can apply this in order to create areas

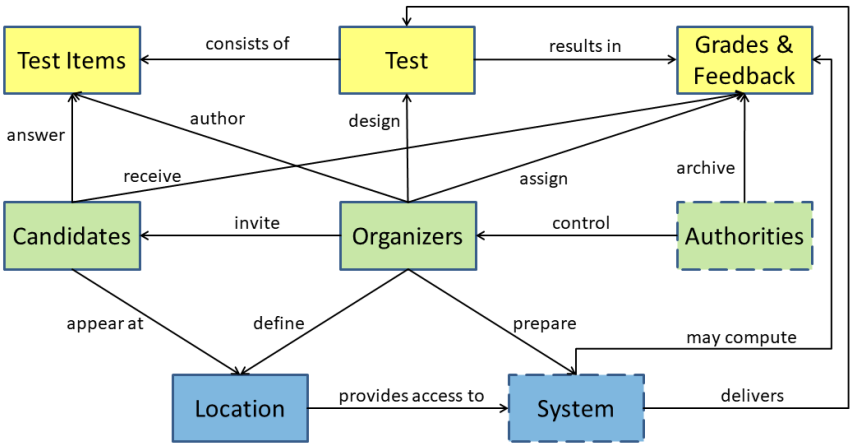


Figure 3.1.: Overview of the eight alphas in the *Kernel for Educational Assessment* and some of their relationships. Optional alphas are shown with a dashed border.

of concern for educational assessment processes in the following way: *People* concerns the alphas representing the actors of the process (i. e. one alpha for each entry from table 2.1 in the previous chapter), *materials* concerns the alphas representing the functional contents reflecting the scientific or professional domain of the assessment (i. e. one alpha for the entries "Question", "Exam" and "Feedback" from table 2.2 in the previous chapter), and *technology* concerns alphas representing the physical way of delivering the assessment (i. e. one alpha for the entries "E-Assessment System" and "Room" from table 2.2 in the previous chapter). The following subsections provide more detailed definitions of the areas and its alphas, while section 3.1.2 will describe and discuss each alpha in detail.

Area of Concern "Content"

Probably the most important parts of an assessment are its functional contents, which are represented by this area of concern. The contents of an assessment reflect its professional or scientific domain. Typically, experts in the particular domain are responsible for creating and maintaining these contents and to assure that they are right. Failure in reaching the desired quality of content in an assessment most likely causes useless assessment results. In general, contents of an assessment are assumed to be independent of the form of the assessment. It is assumed that there is no conceptual difference between assessment contents for an oral exam, a traditional written exam, or an electronic exam. However, the specific presentation of contents may indeed depend on the form of assessment. For example, asking students to draw a complex diagram may be less appropriate for oral exams, while asking students to perform some physical exercise is less usual in written exams.

This area of concern consists of three alphas, representing the different bits an assessment and its results are composed of:

Test Items: The questions and tasks that are potentially used in the assessment. These test items may form a general item pool or several distinct item pools. The name "Test Items" is preferred over other synonyms, as it is concise and corresponds to the name of the next alpha. The name uses the plural form as the alpha refers to the whole amount of test items. Tracking the process of development for individual test items is beyond the scope of a general kernel, but can be added by defining a sub-alpha representing an individual test item.

Test: The actual collection of test items that is delivered to the participants of the assessment. A test may be the same for all participants or may be composed individually from one or more item pools. The name "Test" is preferred over other synonyms, as "Assessment" may be confused with the whole process, "Exam" is inappropriate for formative assessments and "Question Set" may be confused with item pools. The name uses the singular form as it refers to the test as a concept rather than to the test as a particular instance of elements from an item pool. Thus it also covers all variants of a test that may potentially exist. Tracking variants separately is beyond the scope of a general kernel, but can be achieved by adding a sub-alpha "test variant" or similar.

Grades and Feedback: The set of marks, scores, texts or anything else which expresses the result of a test and is used to inform the participants of the assessment about their performance. The name "Grades and Feedback" is preferred over other synonyms to make clear that it includes both textual and numerical descriptions of the participants' performance. A more specific separation can be achieved by adding sub-alphas for grades and textual feedback separately, but as these are usually closely related, it seems useful to stick to a common parent-alpha for them as in this definition.

Area of Concern "People"

The previous area of concern already mentioned domain experts as the authors of assessments contents. As discussed in the summary of section 2.1.3, these are, however, not the people in the focus of an assessment. Hence the focus of this area of concern lays on those people who are more directly concerned with an assessment and who have been identified as the three main actors of assessment processes: The organizers running the assessment (who may of course also author test items as part of their duties while preparing the assessment), the candidates taking part in the assessment and optionally the authorities responsible for the legal aspects of the assessment. If either of these parties fails to fulfill their role within the assessment process, there is no guarantee that the process will produce the desired outcome.

This area of concern consists of three alphas, each representing one of the parties named above:

Organizers: The persons who set up and conduct the assessment. As already discussed earlier, there may be a distinction between people who author test items and design tests and those who use tests in order to assign grades to their students. This alpha makes no assumption on what the actual duties of the organizers are, but assumes a wide

range from the ones named above up to mere organizational ones like inviting candidates, defining time and place of the assessment or preparing e-assessment systems. The name "organizers" is preferred over other synonyms, because it is a neutral term covering the intended broad range of duties. From the alternatives "teachers" is too unusual for institutions other than schools, "examiners" or "authors" are too restrictive in excluding other duties and "faculty" is even more general than "organizers" and thus bears the risk of being too unspecific.

Candidates: The persons who take part in the assessment by answering test items and thus solving a test. The name "candidates" is preferred over other synonyms like "students" or "learners", as these do not necessarily indicate that persons take part in an assessment. Another alternative is "test-takers", which appears to be less usual than "candidates".

Authorities [optional]: The official party that is formally responsible for any legal issues related to conducting the assessment and hence may control the actions of the organizers or archive some of the documents produced during the assessment process. The name "authorities" is preferred over other synonyms, since "exam office" refers less direct to persons as the other alpha names in this area of concern do, while "departmental secretary" is much too specific for a particular organizational structure.

All names use the plural form as each alpha may represent a group of people, where all members of the respective group may or may not have the same duties. For candidates one would typically expect that all of them are treated equally, while organizers or authorities may share their duties among their members. However, it is not the idea of a general kernel to keep track of individuals. Nevertheless, tracking for individual persons or roles can be added by introducing sub-alphas refining the existing alphas. This can in particular be useful if authorities can be split into "internal" authorities of the institution hosting the assessment and "external" authorities like a governmental office. It can also be used to make distinctions between the different duties of organizers that were discussed above. One of the specific sub-alphas can in this case represent the system administrator for an e-assessment system.

Area of Concern "Logistics"

Besides contents and people, there is also a demand for physical or technical facilities to conduct an assessment. In any case, there are one or many physical locations where candidates are located while taking the assessment. Optionally, they are also using a dedicated technical system for doing so, which is the most important aspect which tells a traditional assessment from an e-assessment. The fact that the technical system gets no special handling but is an optional alpha amongst other ones strengthens the idea of seamlessly integrated e-assessment, which is the theme of this publication. It also strengthens the point that the kernel for educational assessment is universal and not limited to e-assessments.

This area of concern consists of two alphas for the physical and technical aspects of assessment organization:

Location: The place or places where the participants are supposed to appear for taking part in the assessment. The name "location" is preferred over the terms from table 2.2, as "room" excludes assessments that happen outside buildings and "physical environment" seems to be too unspecific. The term uses the singular form but also covers multiple places, as these typically need similar treatment. If specific aspects of the location need to be tracked separately (such as preparing a room and moving equipment to that room), sub-alphas can be used.

System [optional]: The electronic system used to conduct the assessment by administering the test items, accepting submissions and associating grades and feedback to submissions. The system may also perform grade and feedback generation automatically. The term "system" is preferred over other synonyms, as it directly refers to the usual term "e-assessment system", while "digital environment" seems to be too general and "tool" seems to be a bit more colloquial. Moreover, "system" naturally includes situations in which several tools are used in conjunction and thus form a complex system. Tracking of individual parts of a system can be added via sub-alphas, but is beyond the scope of a general kernel.

One can imagine to add a third optional alpha for materials needed during the assessment in case the candidates have to perform physical experiments in natural sciences, artistic or musical exercises using instruments or requisites, or similar. However, the states and checkpoints necessary for this kind of alpha are very likely to vary from domain to domain. Thus they are out of scope for a domain independent kernel. Instead, they can be added as domain specific extensions to the kernel, very much like the domain specific extensions that are defined in the original ESSENCE standard for software engineering. Another solution would be to handle them as sub-alphas to "Location" as already mentioned above.

3.1.2. The Alphas

In each of the following subsections, one of the alphas introduced above will be discussed in detail. Following the ESSENCE notation, for each alpha a set of states will be defined as well as a set of checkpoints for each of these states. States and checkpoints will be summarized in a table per alpha, while the text presents the rational for each of these states.

Alpha "Test Items"

A test item is the smallest consistent unit within an assessment that allows candidates to demonstrate their competencies. For the purpose of this kernel, it is assumed that a test item contains some kind of task description and that the candidates are expected to respond to it in some way, e. g. by ticking answer options, writing a short essay, performing some specific activity, or answering orally. Whether answering a test item requires a single competency or a complex set of competencies according to current theories of competency measurement is not relevant here, but discussed later in this publication (see part IV).

The Alpha "Test Items" covers all items potentially used in the assessment and does not ask how an actual test is composed from these items. For practical reasons, the test items may form a general item pool or several distinct item pools from which a certain amount of items is used in the actual assessment. However, it is assumed that all test items that are potentially used need to be prepared in the same way.

A list of states and checkpoints for alpha "Test Items" is provided in table 3.1. The first three states are named "scoped", "designed" and "verified" and are concerned with the different stages of preparation for test items. The alpha particularly reflects the observation that test items have some formal properties (such as an item type, language and intended difficulty) which are defined in the first state, while their functional properties (such as a task description and a sample solution) are defined in the second state. Notably, the name "designed" of the second state does not imply that test items are necessarily designed from scratch right before reaching this state. They can also be drawn from an already existing item bank in which they where inserted earlier. Nevertheless, organizers will first have to define which types and forms of items will be used, before they can "design" them by drawing from an item bank.

As legal regulations may explicitly require a second author to do a review of all proposed test items, the third state handles verification and double-checking. This is also in line with the literature study, in which goal description, authoring and quality assurance appeared as distinct activities. The fourth state of the alpha is named "outcome reviewed" and reflects the didactic practice to review the outcomes of a test with respect to test item performance in order to identify test items with unexpected results (e. g. ones that were often answered wrong by good candidates or ones that were answered right by anybody). This is also an activity found during the literature study.

None of the checkpoints from the four states refers to e-assessments explicitly. However, the form of assessment to be used surely influences that choice of test items to be used. This is reflected by the first two checkpoints of the first state. An e-assessment system may pose some requirements on what types of test items can be used (checkpoint 1) and which form of presentation (textual, graphical, use of special characters, and so on) can be used (checkpoint 2). Notably, it depends on the actual assessment process whether this may stop organizers from using an particular test item: If the decision for using a particular e-assessment system comes early in the process, its requirements with respect to test items may limit the choice. In other cases, the choice of test items to be used can also come first, possibly limiting the choice of systems or forms of assessment that can be used.

Alpha "Test"

A test is the actual collection of test items that is delivered to the candidates of the assessment in some way, e. g. by handing out papers, displaying on a screen or asking questions orally. The alpha refers to the test as an abstract construct and hence does not ask whether a candidate actually sees the whole test at once or only can see and answer the test items within the test one after another. There is also no assumption made on whether the test is a static composition of test items or generated adaptively like in

Test Items: The test items that are potentially used in the assessment. The test items may form a general item pool or several distinct item pools.	
States and Short Description	Checkpoints
Scoped: The types and forms of test items to be used are clear.	• The allowed types of test items to be produced are clear. • The outer form of test items to be produced (e. g. language) is clear. • The intended average size of test items is clear. • The expected characteristics (e. g. relative difficulty) of the test items are clear.
Designed: A sufficient amount of meaningful test items is available.	• There are at least as much test items available as the test consists of. • Texts and any resources belonging to a test item are available for each of them. • A sample solution is available for all test items. • Suitable metrics for computing characteristics of the test items without using them have been applied.
Verified: All test items have been checked to fit the purpose of the assessment and are ready for use.	• It is assured that there is a correct solution to all test items. • It is assured that all sample solutions are correct. • It is assured that there is no unwanted interference between test items. • Test items with unexpected characteristics are eliminated.
Outcome reviewed: Non-optimal items are identified by analyzing test outcomes.	• Suitable metrics for computing characteristics of the test items after using them have been applied. • Test items with unexpected characteristics are identified. • Causes for unexpected characteristics are identified.

Table 3.1.: State and checkpoint overview for alpha “Test Items”

computer adaptive testing. Consequently, a test may be the same for all candidates or may be composed individually from one or more item pools.

A list of states and checkpoints for alpha “Test” is provided in table 3.2. The first and second state are named “goals clarified” and “designed” and correspond to the first two states of the alpha for test items, as also the whole test needs both a definition of its formal and functional properties. The third state is named “generated” and is fulfilled when an actual instance of the test is created for each candidate. As already mentioned above, this may be a physical representation such as some pieces of paper, but it may also be the specific sequence of questions asked to one particular candidate in an oral exam. The fourth state is named “conducted” and is fulfilled when all candidates have completed their tests. Notably, in a written exam this state may be reached days or even weeks after “generated” (depending on how long before the day of the test the exam sheets are printed), while in an oral exam it may be reached minutes or even seconds

Test: The actual collection of test items that is delivered to the candidates of the assessment. A test may be the same for all candidates or may be composed individually from one or more item pools.	
States and Short Description	Checkpoints
Goals clarified: The purpose of the assessment is clear.	• It is clear whether the assessment is diagnostic, formative or summative. • The relevant topics for the assessment are clear. • The expected characteristics of the test are clear.
Designed: The size and content of the test as well as its outer form of presentation is defined.	• A set of potential test items for each topic in the test exists. • The number of test items to be included in the test is clear. • The mode of choosing test items for an actual test instance is clear. • It is clear how the test items will be presented to the participants and how submissions will be collected.
Generated: Actual instances of the test are available for each candidate.	• Individual test items are available for each candidate. • It is clear which test items an individual participant has to answer to. • The outer form of presentation for the test has been produced. • It is assured that each individual test conforms to the goals of the assessment.
Conducted: All submissions from all candidates are collected.	• All participants have made a submission or explicitly stated that they don't want to make one. • There are no submissions left that are not yet collected.
Evaluated: Grades and feedback have been assigned to all submissions.	• All submissions have been evaluated. • Grades and feedback have been generated for each submission. • Suitable metrics are applied to analyze the actual characteristics of the test.

Table 3.2.: State and checkpoint overview for alpha "Test"

after the last question is posed. The fifth state represents the fact that a test needs to be evaluated and also includes the retrospective analysis of test item performance as above.

The alpha makes no explicit reference to e-assessment in any of its checkpoints, but several implicit ones. In state "Designed", both the mode of choosing test items (checkpoint 3) and the way of presenting test items (checkpoint 4) may be subject to requirements from an e-assessment system. As already discussed for the previous alpha, an early decision for a particular e-assessment system may limit the choice here, while a late choice of an e-assessment system is limited by the decisions made during test design. In state "Generated", at least the production of the outer form of presentation (checkpoint 3) relates to e-assessment systems, as this is one of their primary duties. Similar is true for the generation of grades and feedback (checkpoint 2 of state "Evaluated"), but only in case the e-assessment system supports automated grading, which is not necessarily true

for complex exercises. Hence the number of checkpoints in this alpha that are influenced by conducting an e-assessment instead of a non-electronic assessment clearly depends on the capabilities of the e-assessment system used.

Alpha "Grades and Feedback"

Each response to a test item can be evaluated in order to produce the actual test result. Depending on the didactic setting of the assessment, these results may consist of marks, scores, credit points, texts or anything else which is used to inform the participants of the assessment about their performance. Results can be assigned both to single test items and to the whole test (or arbitrary parts of it). The alpha covers all these different kinds of feedback and makes no assumption on whether participants have access to results during the assessment or only afterwards.

A list of states and checkpoints for alpha "Grades and Feedback" is provided in table 3.3. Again, the first two states are concerned with preparations: State "granularity decided" reflects the fact that there are many ways of how to give feedback and that the didactic purpose of the assessment determines the choice. State "prepared" refers to the creation of appropriate marking schemes or alike as well as organizational set-up of grading sessions. The preparation of grading schemas is also one of the activities found explicitly in the literature study. The third state is named "generated" and is fulfilled if all grades and feedback are created. Grading answers and marking tests was the activity with the most occurrences in the literature study.

The final state is fulfilled when grades and feedback are available to the candidates and is thus named "published". Notably, in a written exam it may take some time after the submission to reach state "generated" and it may also take some more time to reach "published", while in an oral exam feedback is often generated right after a candidate answered a question and is also published immediately by responding to the candidate's answer. However, as the alpha refers to grades and feedback in general, state "published" may nevertheless be fulfilled later, as grades are typically not mentioned after every answer, but only at the end of an exam or even at some later point in time. In the literature study the reporting of grades has been identified as separate activity, which also motivates to make a distinction between the generation of grades and its reporting.

Again there are no explicit but some implicit references from the checkpoints of the states to the use of e-assessment systems. First, the kind of grades and style of feedback (checkpoints 1 and 2 in state "Granularity decided") can depend on the e-assessment system, if it is used to generate or present grades and feedback. The preparation of marking schemas (checkpoint 1 in state "Prepared") is a crucial point when using automated grading features of e-assessment systems, as grading and feedback rules need precise definitions in this case. In manually marked assessments, marking schemas may omit some corner cases, which will then be discussed and solved individually later if they actually occur. This may not be possible when using an e-assessment system which will consequently require more effort to reach state "Prepared". On the other hand, the resource allocation for grading mentioned in the same state (checkpoint 3) can be easier when using an e-assessment system with automated grading capabilities. In state

Grades and Feedback: The set of marks, credit points, texts or anything else which is used to inform the participants of the assessment about their performance.	
States and Short Description	Checkpoints
Granularity decided: The kind and style of grades and feedback is clear.	• It is clear which kind of grades will be used. • It is clear which style of feedback will be used. • It is assured that the kind and style of grades and feedback fit the purpose of the assessment.
Prepared: Criteria are clear for when to generate which grades and feedbacks.	• Proper marking schemas, rubrics or alike are available. • Resources are allocated to review submissions in order to create grades and feedback. • The time frame for creating grades and feedback is clear. • The risk of solving the test by guessing is under control.
Generated: Grades and feedback for all submissions are created and assigned.	• All submissions have been reviewed. • Grades and feedback are created for all submissions. • Grades and feedback have been checked to be assigned correctly to submissions.
Published: Candidates have access to grades and feedback for their submissions.	• Grades and feedback are available for review by the candidates. • The time frame and process for placing complaints is clear.

Table 3.3.: State and checkpoint overview for alpha "Grades and Feedback"

"Generated" the double-checking of generated grades and feedback (checkpoint 3) can gain specific legal relevance when using e-assessment systems with automated grading capabilities. Depending on the laws applicable in a specific case, it may be necessary that each and every feedback that was generated automatically must be reviewed manually before it gets published. However, laws may also require double-checking for manual grades but not for automated grades, if feedback rules and grading schemas have been double-checked before.

Alpha "Organizers"

For each assessment there is at least one person responsible for organizing it and thus managing the assessment process. For larger assessments it can be assumed that more people are involved in setting up and conducting the assessment, including test item authors, assessors and technical staff. Each of them picks up parts of the responsibility for conducting the assessment and is thus responsible for some part of the assessment process. People how just assist in conducting the assessment but do not have a final responsibility with respect to the process are not considered as organizers, but are included indirectly by mentioning them in the checklists.

Organizers: The persons who set up and conduct the assessment.	
States and Short Description	Checkpoints
Identified: It is clear which persons need to be involved in organizing the assessment.	• The required persons are known by name and role. • It is clear how to contact and involve the persons.
Working: Organizers have picked up their individual tasks.	• Organizers know their individual role and duty in the assessment. • Organizers know what they have to prepare for the start of the assessment. • Organizers know the deadlines for their individual tasks.
Satisfied for Start: Organizers are ready to start the assessment.	• Organizers agree that preparation is successfully finished. • Organizers or their representatives are available for monitoring the assessment.
Satisfied for Closing: Organizers have no more open duties.	• All post-processing of the assessment by the organizers has been completed. • Didactic evaluation of the assessment has been done. • Hints for improving future assessments have been documented.

Table 3.4.: State and checkpoint overview for alpha “Organizers”

A list of states and checkpoints for alpha “Organizers” is provided in table 3.4. The first state is named “identified” and thus represents the fact that it may require some work to find out who needs to be involved into the assessment for which tasks. The second state is named “working” and is fulfilled when all responsible persons have picked up their duties. This also strengthens the use of processes as means of coordinating people. Once they have done everything that is required to start the actual assessment, state “satisfied for start” is reached. Similarly, the final state “satisfied for closing” is reached when all evaluation and post-processing is done and the organizers have no more open duties. Neither of these states can directly be deduced from the activities identified in the literature study. Instead, these states follow the rational the states of alphas “Stakeholders” and “Team” from the ESSENCE Kernel of Software Engineering.

Different to the alphas discussed so far, there is only an implicit relationship between the states and checkpoints of this alpha and the use of an e-assessment system. This relationship is established by the fact that a system administrator for an e-assessment system may be counted among the organizers of the assessment. However, none of the checkpoints of this alpha gains new characteristics due to that choice.

Alpha “Candidates”

The largest group of people concerned with an assessment are usually the candidates, which are the persons who take part in the assessment by answering a test. Although

they are involved personally in the assessment process for a relatively short period of time, the proposed kernel includes an alpha with seven states to represent all essential aspects related to candidates.

A list of states and checkpoints for alpha "Candidates" is provided in table 3.5. The first two states are named "scoped" and "selected" and refer to the part of the process in which it is first defined who is allowed to take part in the assessment and secondly the actual persons are identified. Both states may belong to activity "register students" as found in the literature study. The third state "invited" is fulfilled when candidates know how to prepare themselves for the assessment.

The following two states "present" and "dismissed" refer to the physical presence of the candidate at the location where the assessment takes place. Notably, that does not mean that all candidates will be at the same place at the same point in time. They are also considered "present" if they are in different locations and it is also possible that some candidates are already dismissed, before the last one is present, as it is usual in oral exams. The sixth and seventh state are named "informed" and "satisfied" and reflect the fact that candidates need explicitly to be informed about their results (which corresponds to state "published" for grades and feedback) and then often have some time frame to place complaints before the grades formally count as accepted. The latter has also been identified as an activity in the literature study.

None of the states of this alpha has any specific relationship to using an e-assessment system, neither explicitly nor implicitly. Candidates always have to respond to test items in a specific way in any form of assessment, so using an e-assessment system is only one way among others and does not change the process from the candidates' point of view. The same is true for review of results, which may or may not happen using an electronic system.

Optional Alpha "Authorities"

Depending on the didactic and formal setting of the assessment some official party may be formally responsible for any legal issues related to conducting the assessment. As this may introduce additional process steps or dependencies between states, authorities are introduced as an additional optional alpha in the kernel. This alpha is only relevant for formal assessments.

A list of states and checkpoints for alpha "Authorities" is provided in table 3.6. The first state is named "Identified" and covers the same aspects as the corresponding state of alpha "Organizers". The second state is named "Involved" and is fulfilled when all assessment information relevant to the authorities have been provided. The naming of the state is different from the second state of alpha "Organizers", as authorities are supposed to play a less active role in the assessment process. Hence they may be involved in terms of providing information or verifying documents, but do not necessarily work in terms of creating contents or making design decisions.

The third and fourth state are named "Satisfied for Start" and "Satisfied for Closing", which is again similar to the states of alpha "Organizers". They are reached when there

Candidates: The persons who take part in the assessment by solving a test.	
States and Short Description	Checkpoints
Scoped: The criteria for being allowed to take part in the assessment are clear.	• Criteria exist on how to select candidates for the assessment. • The criteria fit the purpose of the assessment. • It is clear how to apply the criteria in order to identify relevant persons.
Selected: The individual persons who are allowed to take part in the assessment are identified.	• Individual persons allowed to take part in the assessment are identified by name or some other proper measure. • It is clear why these persons are allowed to take part in the assessment. • It is clear why no other persons are allowed to take part in the assessment.
Invited: The persons who are allowed to take part in the assessment are informed about all relevant circumstances of the planned assessment.	• Candidates know where and when they are expected to appear. • Candidates know what they must and must not bring along for the assessment. • Candidates know about any other relevant circumstances of the planned assessment.
Present: The persons who are allowed to take part in the assessment are present at the assessment location.	• Candidates are physically present where they are expected to be. • It is clear that these persons are indeed the ones that were invited to take part in the assessment. • It is made sure that no unauthorized person is present.
Dismissed: The persons who took part in the assessment have finished their on-site duties.	• Candidates have submitted their solutions. • Candidates have left the assessment location.
Informed: The persons who took part in the assessment know their results.	• Candidates know the grades assigned to their submissions. • Candidates know any feedback that was intended to be directed to them. • Candidates know the procedure for placing complaints.
Satisfied: The persons who took part in the assessment agreed that the assessment results are valid.	• Candidates had a suitable opportunity to review grades and feedback. • Candidates had a suitable opportunity to ask questions about their results and articulate disagreement. • Any conflicts are resolved.

Table 3.5.: State and checkpoint overview for alpha “Candidates”

Authorities: The official party that is formally responsible for any legal issues related to conducting the assessment. This alpha is only relevant for formal assessments.	
States and Short Description	Checkpoints
Identified: It is clear which authorities need to be involved in the assessment.	• The required authorities are known by name and role. • It is clear how to contact and involve the authorities.
Involved: Authorities have access to assessment information relevant to them.	• Each authority is represented in the assessment process by an appropriate person. • Authorities had a suitable opportunity to place their questions and requests about the assessment process. • Assessment organizers have targeted their questions towards the authorities.
Satisfied for Start: There are no more legal obstacles to start the assessment.	• All required documents about the preparation of the assessment are available and complete. • Permissions from the authorities are present.
Satisfied for Closing: All legal files for the assessment are complete and ready to be closed.	• All required documents about the actual process of the assessment are available and complete. • All required documents about the results of the assessment are available and complete.

Table 3.6.: State and checkpoint overview for alpha “Authorities”

are no more legal obstacles to start the assessment or the legal files for the assessment are ready to be closed, respectively.

As the authorities are considered to be less actively involved in the process compared to the organizers, they also consider the assessment process from an more abstract perspective. Consequently, they have no implicit or explicit relationship to the potential use of e-assessment systems. The required documents that have to be prepared or reviewed by the authorities may differ between electronic and non-electronic assessments, but that does not change the characteristics or importance of the checkpoints and states.

Alpha “Location”

It is assumed that each assessment needs some physical location where candidates will be located while taking part in the assessment. Depending on the kind of size of assessment, this may be a single room for all candidates (at the same time or one single candidate or group after another) or a set of distributed locations. As no further assumptions are made by this alpha, this may also cover for example field tests during excursions or informal assessment situations in which candidates take their tests on a mobile device while sitting in a bus or train.

A list of states and checkpoints for alpha “Location” is provided in table 3.7. Quite similar to the states for candidates, the first two states for the location are named

"defined" and "selected" and refer to the fact that first some abstract requirements are formulated towards the properties of the assessment location and then an actual room or set of rooms is selected. As rooms are physical resources that may cause conflicts with other assessments happening at the same time, state "reserved" is explicitly introduced to cover the necessary communication and also the calculation of set-up time. In the literature study, these states were at least partially covered by the activity of scheduling an assessment.

If all set-up is done, the location is considered "prepared", which is the fourth state corresponding to "satisfied for start" for the organizers and also to the respective activity identified in the literature study. The final two states are named "in use" and "left" and correspond to some extent to "present" and "dismissed" for the candidates but also cover the fact that the location needs to be restored after the assessment.

Since the assessment location is the physical place where candidates come in contact with the assessment, it may also establish relationships to an e-assessment system. In state "Defined" the required technical equipment (checkpoint 2) directly depends on whether an e-assessment system is used or not. Typically, this decision will be made first and thus pose requirements towards the location. The opposite case in which the location is decided first and the decision on whether an e-assessment system is used or not comes second (and is constrained by the choice of location) can be considered to occur very rarely. Which technical equipment is required depends on the specific mode of electronic assessment and is beyond the discussion of a general kernel. On can imagine both fully-equipped computer pools as well as just providing wireless network access so that candidates can use their personal mobile devices.

In state "Reserved", both the planning of preparations (checkpoint 2) and the planning of clean-up efforts (checkpoint 3) possibly refer to the use of e-assessment system and the required time may be very different for electronic assessments compared to non-electronic ones. In the remaining states, any mentioning of technical equipment may relate to the use of e-assessment systems as already discussed for the first state.

Optional Alpha "System"

In case a computer-aided assessment system or similar electronic system is used to conduct the assessment, it can be represented by an additional optional alpha. The alpha covers all possible duties of this system such as administering the tests, accepting submissions, associating grades and feedback to submissions and performing grade and feedback generation automatically. This alpha is only relevant for electronic assessments.

A list of states and checkpoints for alpha "System" is provided in table 3.8. Similar to the previous alpha, the first two states are named "Defined" and "Selected". This again reflects the fact that (at least in an ideal scenario) one would first define some abstract requirements towards the assessment system and then select and actual system fulfilling these requirements. In reality, organizers sometimes have no choice, as they must use the system provided by their institution. In this case, these two states are fulfilled by default and the features of the available system may restrict organizers in the selection of test item formats they can use. Since the ESSENCE notation does not require to

Location: The place(s) where the candidates will be located while taking part in the assessment.	
States and Short Description	Checkpoints
Defined: The requirements towards a proper location for conducting the assessment are clear.	• The required overall size of the location is clear. • The required technical equipment in the location is clear. • Any requirements regarding reachability and accessibility of the location are clear.
Selected: It is clear which actual location is the preferred one that fulfills all requirements.	• An actual physical location is known by room name (or alike) that fulfills all requirements. • An alternative location is possibly also known.
Reserved: It is assured that the location is available for all participants at the time of assessment.	• It is clear to all relevant persons that the location will be used for assessment at the planned time. • It is clear which preparations are pending and how much setup time they require. • It is clear which cleanup effort is required after the assessment and how much time it requires.
Prepared: The location and all required facilities are set up and ready for use.	• The required technical equipment or materials at the location are ready for use. • The location is reachable and accessible for all candidates. • A sufficient amount of organizers or their representatives is present at the location.
In Use: The candidates have taken their seats at the location.	• The location is populated with candidates. • Technical equipment or materials are in use. • Proper use of the location by the candidates is monitored.
Left: The candidates have left the location and everything is restored.	• No candidates or organizers are left at the location. • All technical equipment or materials used during the assessment are restored or removed. • All cleanup is done.

Table 3.7.: State and checkpoint overview for alpha “Location”

define dependencies between states from different alphas explicitly, processes for both orders can be defined and monitored using this kernel. As already discussed earlier, this also points out that from a process point of view the introduction of a computer-aided assessment system does not necessarily restrict organizers in the freedom of test design.

The third state is named “Available” and refers to the fact that the selected system also needs to be accessible to continue preparation. This in turn will lead to the fourth state, which is named “Ready for Start”. The fifth state is named “In Use” and depicts the period of time in which candidates interact with the system and also the period of time in which it performs tasks like automated grading on its own. The final state is named “Ready for Closing”. The state makes no assumptions on whether the whole

system will actually be closed or whether it is just the assessment that is closed and archived. However, it is assumed that any remaining steps of the process will not require any more interaction with the assessment system.

3.1.3. The Activity Spaces

Defining and using activity spaces is not necessary in any case, as processes can also be defined just on top of alphas and alpha states. However, the definition of alphas and states as performed so far did not systematically cover all activities summarized in table 2.3 in section 2.1.3. Hence it can be considered useful to include activity spaces into the Kernel of Educational Assessment as the ESSENCE standard does in the Kernel for Software Engineering. Some of the activity spaces can directly be derived from the activities found during the literature study. On the other hand, the definition of alphas and states was not entirely based on the identified activities, but also reused concepts from the ESSENCE Kernel for Software Engineering. Hence it must also be considered to refer to some of the activity spaces defined there. This applies in particular to the area of concern "People", as its alphas also used some of the alpha states known from the ESSENCE Kernel for Software Engineering.

The definition and discussion of the Activity Spaces is again organized according to the three areas of concern.

Area of Concern "Content"

The activities concerned with handling the contents of an assessment can be grouped into four activity spaces:

Scope the Assessment: Two activities have been identified in the literature study that are concerned with defining the goals of the assessment independent of the concrete content. Besides direct goal definitions, also the definition of a grading schema or at least the definition of thresholds for pass and fail have to be counted here, as these are typically defined independent from the actual assessment contents. Moreover, general decisions like deciding on the language or duration of the assessment can be considered as part of this activity space, although not identified explicitly in the literature study. The activities within this activity space may be constrained by previous decisions on using an e-assessment system or they may constrain the later choice of an appropriate way of delivering the assessment.

Create the Assessment: Creation of assessment involved various activities concerned with authoring of items and tests as well as with verification and double-checking. These activities can typically only be started when the scope of the assessment is clear and must in turn be completed before the assessment can be conducted, so this defines a separate activity space. If an e-assessment is used, activities within this activity space will involve interaction with the e-assessment system.

Conduct the Assessment: Several activities have been identified in the literature study that are concerned with the core steps for conducting the assessment, such as distribution and collection of tests, answering to test items or monitoring the assessment.

System: The electronic system used to conduct the assessment by administering the tests, accepting submissions and associating grades and feedback to submissions. The system may also perform grade and feedback generation automatically. This alpha is only relevant for electronic assessments.	
States and Short Description	Checkpoints
Defined: The requirements towards a proper system for conducting the assessment are clear.	• Required test item types and ways of submission are clear. • Required grading and feedback generation features are clear. • Required management features are clear. • Requirements regarding access to the system are clear.
Selected: It is clear which actual system is the preferred one that fulfills all requirements.	• A system fulfilling all requirements is known by name. • It is clear how to proceed to make the selected system available. • It is clear whether training on the system is necessary for any of its users.
Available: It is assured that the system is available for preparing and conducting the assessment.	• Organizers can access the system for preparation. • Organizers know how to use the system. • It is assured that candidates will be able to access the system during the assessment.
Ready for Start: All system settings are ready to start the assessment.	• Test items are available in the system. • Required grading and feedback generation features are set up. • A dummy assessment has successfully been conducted on the system. • Candidates know how to use the system.
In Use: The system is used by the candidates of the assessment or is running automated tasks.	• The system is accessible to all candidates. • Tests are delivered by the system. • Submissions are recorded by the system. • Automated tasks are performed by the system. • Critical system behavior is monitored or recorded for further inspection.
Ready for Closing: The system is ready to be shut down.	• No more submissions are made to the system. • No more grading and feedback generation tasks are open on the system. • Any data required for future access is exported from the system. • Any suspicious system behaviour has been documented.

Table 3.8.: State and checkpoint overview for alpha “System”

In contrast to the other activity spaces, activities in this one typically only take a short period of time, as conducting an assessment usually only takes some hours or even less, while preparations or grading can take days or even weeks. Consequently, interruptions for activities in the other activity spaces are not as critical as they can be in this one. Similar to the previous activity space, activities in this one will also involve interaction with the e-assessment system in case of electronic assessments.

Post-process the Assessment: This activity space contains all activities to be performed after all answers from the candidates have been received. Besides the actual task of grading answers in order to assign feedback, the literature study also identified checking for plagiarism as another potential activity in the post-processing of assessments. Also the analysis of the overall assessment result in order to improve the next iteration of the assessment is part of this activity space. Even in case of electronic assessments, the activities within this activity space may require little interaction with an e-assessment system. It very much depends on the features of a system whether grading is an automated or manual task within that system or whether submissions are exported and graded externally.

Area of Concern "People"

The activities related to managing people involved in the assessment process can be grouped into three activity spaces:

Identify Stakeholders: All three alphas in this area of concern had starting states concerned with identifying that actual persons that are involved in the assessment. Following the terms from the ESSENCE standard we can refer to them as the stakeholders of the assessment. Identifying them seems to be an important effort according to the states of the alphas and hence justifies to define an activity space for the necessary activities. With "register students" at least one activity belonging to this space has been identified in the literature study. When using an e-assessment system, a particular activity in this activity space can be to identify the responsible system administrator. This activity can be omitted if no e-assessment system is used.

Coordinate People: The literature study also revealed that communication among the stakeholders of an assessment is a crucial point. This is also reflected by the fact that each alpha in this area of concern has at least one state that checks that the respective stakeholder is involved, knows their duties and also has enough information to fulfill their duties. Consequently, we can assume a lot of communication activities filling this activity space. During the literature study, staff allocation was found as one example for activities filling this activity space. Similarly, activities related to communication with system administrators may be placed in this activity space in case of electronic assessments.

Satisfy Stakeholders: This activity space complements the post-processing of assessments from the "content" area of concern in that it is concerned with the dissemination of the results produced during post-processing. As particular activities in this activity space the literature review identified the reporting of grades and the handling of appeals during the review of results. Even when using an e-assessment system there may be

no need for specific activities related to that system within this activity space, if no extensive post-processing for system clean-up or archiving the results is required.

Area of Concern "Logistics"

The activities related to the handling and preparation of the physical and technical environment of an assessment can be grouped into four activity spaces:

Plan Logistics: Both alphas in this area of concern start with states that tackle planning of logistics in terms of defining and selecting the required resources. Hence we can also assume that there is a relevant amount of activities that can be grouped into this activity space, although only one activity on scheduling the assessment has been identified explicitly in the literature study. Using an e-assessment system will contribute specific activities to this activity space.

Prepare the Environment: The preparation of the environment was also only reflected by one activity in the literature study, but it can be assumed that more activities will be necessary depending on the domain and thus the kind of assessment. Notably, activity spaces are defined in the kernel, but may be filled by domain specific extensions. As mentioned earlier, one could think of a third alpha representing physical materials needed during the exam and the necessary activities to prepare those would also fit into this activity space. Also preparing an e-assessment system may add specific activities to this activity space in case of electronic assessments.

Operate and Monitor the Environment: The same as before is also true for this activity space, as the environment may need no specific monitoring in the general case, but may also need special attention in some domains. For e-assessments it can be at least considered necessary to monitor system operation and network connectivity.

Restore the Environment: Similar to the preparation of the assessment environment it may also be necessary to restore things that have been set up specifically for the assessment. This applies as well to electronic systems, in which it may be necessary to remove data no longer needed for the purpose of the assessment.

3.2. Process Phases of Educational Assessment

While the kernel defines alphas independent both of the domain and the process they are used in, there is also a need to identify a generic model for process descriptions based on this kernel. The activity spaces in the kernel cannot be considered sufficient here, as they only provide chronologically ordered slots to group activities, but they are not intended to express dependencies between activities or alpha states from different areas of concern. However, both strict process definitions in terms of sequences of activities and more agile process definitions in terms of milestones need to relate activities or alpha states from different areas of concern to each other.

One way of expressing these relationships and thus describing processes based on alphas is to chain alpha states in the order they have to be reached. This chaining can be done without additional information or by defining activities that list the actual things to do while proceeding from one state to another. However, this is a very strict

way of modelling a process and does not allow for more abstract definitions or the agile enactment of processes. Hence a more generic model of process phases is used here, which is also illustrated by example in the appendix of the original ESSENCE specification. This model groups states from several alphas into one phase and defines the process as a linear sequence of phases. One phase can cover more than one state of a single alpha, but there may also be alphas that do not contribute one of their states for a particular phase. In terms of the original ESSENCE standard for software engineering, these phases can be expressed as patterns, which are generic constructs that relate some kernel elements to each other. The idea of using phases as a means of structuring a process model has also been used in some of the papers from the literature study (i. e. [310, 170, 193]).

For the cases studies presented in the next chapter, up to five different phases are used and thus described in the following. Neither of them has to be considered mandatory for assessment process descriptions. Similarly, a process description may also add an additional phase if necessary.

3.2.1. Phase "Planning"

The planning phase is intended to contain all the alpha states that will be considered while creating a conceptual plan for the assessment. While scope, shape or the number of people involved in the assessment are not clear at the beginning of this phase, most of these bounds and circumstances should be made clear during this phase. However, alpha states dealing with details that are considered of minor importance in the actual process can be deferred to later phases. On the other hand, any state bearing major decisions about cancelling the assessment should be included into this phase, as cancelling later will result in wasting significant amounts of work.

3.2.2. Phase "Construction"

The construction phase is concerned with all alpha states that relate in some way to the production of resources and artifacts needed during the assessment. It also is intended to assemble states concerned with the organizational, legal, or physical setup of prerequisites for conducting the assessment. Any state that is considered to be completed before the actual assessment can start should be placed in this phase at the latest. It can be assumed that a significant amount of time will be spent on tasks arising from this phase.

For small processes it can be meaningful to combine planning and construction phase into one if activities in there happen more interwoven and less bureaucratic. In any case, both phases are concerned with the preparation of the assessment.

3.2.3. Phase "Conduction"

The conduction phase represents the – possibly very short – time frame in which the actual assessment is conducted. Thus all states related to delivering tests, collecting submissions and monitoring the assessment should be grouped in this phase. In particular, this is most likely the only phase in which all of the candidates have direct contact with the assessment.

For assessments that are created in an "on the fly" manner caused by direct interaction between candidate and assessor it can be meaningful to combine construction and conduction phase into one. This would imply that the actual contents of the assessment are not entirely known at its beginning, but are created spontaneously. This may be particularly suitable for assessments in extensively creative domains in which unplanned events are likely to occur, but also in any other kind of assessment in which the organizers wish to interact very closely with the candidates. It is likely that some states that are considered to be part of the construction phase in other processes are moved to the planning phase in these cases.

3.2.4. Phase "Evaluation"

The evaluation phase is concerned with all activities related to assessing submissions or answers from the candidates and generating feedback. Hence all states that are related to the evaluation of the candidates' performance should be grouped in this phase. From the didactic point of view this is one of the most important phases, as this phase produces the actual outcome of the assessment and thus contributes much to its overall value. Depending on the domain of the assessment, the test item types used and the mechanisms applied for grading, this phase can consume a lot of time in the whole assessment process.

From a technical perspective, one can distinguish between synchronous and asynchronous feedback generation or grading. In the latter case, feedback is generated at some point in time after submitting an answer to the test item, while in the former case feedback is generated as soon as an answer is submitted. In particular, asynchronous feedback generation can defer grading until all answers from a single candidate or all answers from all candidates have been handed in. On the other hand, synchronous grading can be used to let the next test item depend on the correctness of the answer to the previous one. This also has consequences on the design of the assessment process: If only asynchronous grading is used, conduction and evaluation can form two distinct phases. If synchronous grading is used at least in some cases, conduction and evaluation happen in parallel and thus form a joined phase.

3.2.5. Phase "Review"

The review phase is considered to be the final phase in the assessment process. It is intended to cover all tasks the remain after the assessment is both conducted and evaluated. Thus it should cover both legal and organizational post-processing and also tasks on documenting how well the assessment process actually worked. It is likely that some people who have been involved in the assessment process so far have no duties in this phase and thus can leave the process early. Consequently, some alphas may have reached their final state already in an earlier phase and do thus not contribute to the review phase.

4. Case Studies

This chapter illustrates how to use the kernel and process phases defined so far and thus demonstrates that they are indeed suitable to model the process of different kinds of educational assessments. We will look on four examples that represent assessments from different contexts. The resulting process descriptions vary both in the number and kind of alphas as well as phases that are used to describe the processes. After presenting and discussing the phase model for each case, there is also a discussion of activities occurring in the respective process and on how they fit into the activity spaces defined in the kernel. Additionally, we will discuss which work products are likely to occur in the process. Both the discussion on activities and the discussion on work products are not meant to be exhaustive, but serve as illustrations on how to add information in an actual process beyond the concepts taken from the kernel.

4.1. Case 1: A Traditional Oral Exam

The first case study considers a traditional oral exam, in which an assessor asks questions to a single candidate who in turn answers these questions immediately.

With respect to the kernel and process phases defined in the previous chapter, this case has two remarkable characteristics: First, it does not make use of any electronic system and hence we do not need to consider the alpha "System" in our process description. Second, during the assessment the assessor will iteratively pose questions, evaluate the answer and generate new questions. Consequently, conduction and evaluation of the assessment can be considered to form one joined phase. One could even think of joining this phase with the construction phase, because an assessor may be able to create new questions during the exam. However, it is very likely that an assessor will have some amount of "standard" questions constructed earlier as well as some kind of plan on which topics to handle in which order during the exam. As these are arguments in favor of a separate construction phase, there is one in this case study. Another option would be to join planning and construction phase, which is nevertheless not examined in this case study. The resulting process description in terms of alpha states assigned to phases is depicted in figure 4.1.

Following the general reasoning of the process phases, the first phase contains only the first state for each of the alphas, as it is concerned with planning but not with actual preparations. Some of the checkpoints of these states may be fulfilled right from the beginning, such as the language used in the assessment or the organizers that are involved. Depending on the habits in a particular institution it may also happen that some checkpoints or even states from the next phase are also fulfilled right from the beginning. This may in particular be true for the alpha "Location", as oral exams often

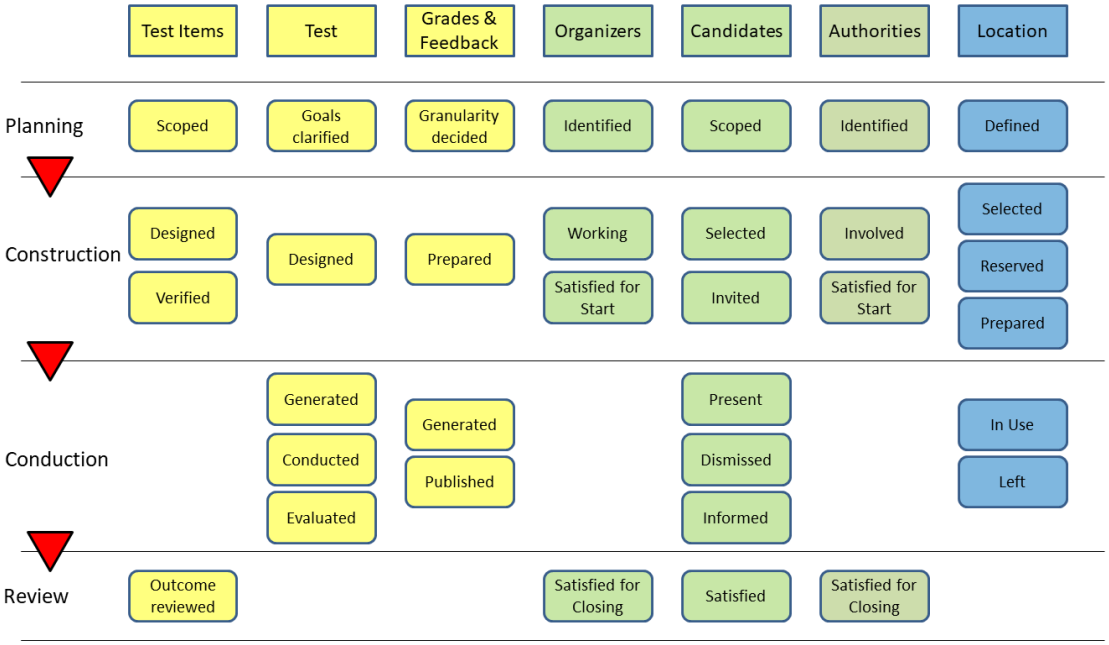


Figure 4.1.: Overview on the assessment process for a traditional oral exam using four phases. As this exam is no e-assessment, the alpha "System" is not used in the process description.

take place in the assessor's office, which needs no specific reservation. However, there is no immediate need to shift the respective states to the first phase for that reason, as the ESSENCE standard allows for agile enactment of processes and thus does not forbid to have some states in one phase already fulfilled before other states in a previous phase are fulfilled as well.

The construction phase also follows that general reasoning of the process phases and also has contributions from all seven alphas in the process. When all states in this phase are fulfilled, everything is ready to start the actual assessment. The reasoning for joining conduction and evaluation into one phase was already discussed above. Notably, only four out of the seven alphas contributed to that joined phase. This is not surprising, as test items, organizers and authorities are not supposed to change their state while the assessment is conducted. From the four alphas contributing to this phase, three also reach their final state in this phase. This also matches the expectation that test, grades and feedback, and location will not change their state after the assessment has been conducted and evaluated. Hence the remaining review phase only contains the final states of the remaining alphas. It depends again on the habits within a particular institution whether the final state for alpha "candidates" can also be shifted to the third phase. This is the case when all discussion about the final grade is done within the assessment session of the individual candidates. If – from a legal point of view – leaving the assessment

location means to accept the final grade, then this state can be moved. However, as this is not the general case for oral exams, it is placed in the fourth phase in this case study.

Following the idea of the ESSENCE standard, one can enrich this process description by naming work products referring to alphas and activities filling the abstract activity spaces. Considering the former, a relatively small amount of work products is involved in the process of an oral exam. Most likely, a list of candidates will be used that also contains information about the schedule. Potentially the same list or at least a very similar one can be used to report grades to the authorities. For legal reasons, writing a log record for each assessment may be mandatory, which would be another kind of work product. Concerning the activity spaces, the space for creating the assessment may contain only very few activities, as creation of physical artifacts such as exam sheets is not necessary. In turn, the spaces on coordinating people and satisfying stakeholders will involve activities related to communication with the exam authorities. The spaces in the logistics area of concern may in general contain less activities, as oral exams are often conducted in a normal office and thus need only few specific preparations.

4.2. Case 2: A Summative E-Assessment

The second case study considers a summative e-assessment such as an electronic exam. It is assumed that candidates come to the exam hall which is equipped with computers and appropriate systems for the purpose of publishing the test and collecting submission. Furthermore it is assumed that there is no need to provide direct feedback to the candidates while they are present in the exam hall. Grading of the solutions can thus happen asynchronously. Hence with respect to kernel and process phases, this case has the following significant characteristics: First, all alphas including the optional ones need to be used, as we employ an electronic system and have to involve the exam authorities. Second, we can use all five phases suggested in the previous chapter, as we can clearly separate the conduction phase from the evaluation phase. The resulting process description in terms of alpha states assigned to phases is depicted in figure 4.2.

The first phase does not differ from the one shown in case study 1, as the same argumentation applies here as well. However, in the case of a summative e-assessment it is less likely that checkpoints from alpha “Location” in later phases are already fulfilled right from the beginning. Instead, this may be true for alpha “Test Items”, if the assessment is based on a predefined item pool. This is no particular feature of e-assessments, but also applies to traditional written exams in which existing item sets are reused. Nevertheless this case study assumes a less specific case and thus handles preparation of the item pool among other states in the second phase. Different to case study 1, state “Generated” of alpha “Test” is also placed in the construction phase, as it is assumed that the test is not created dynamically for each individual candidate in this scenario. To handle such e-assessments, the state needs to be moved to the conduction phase as case study 3 will show.

The conduction phase has only contributions from four out of the eight alphas in the process. The same arguments as in case study 1 apply here, as test items, grades and

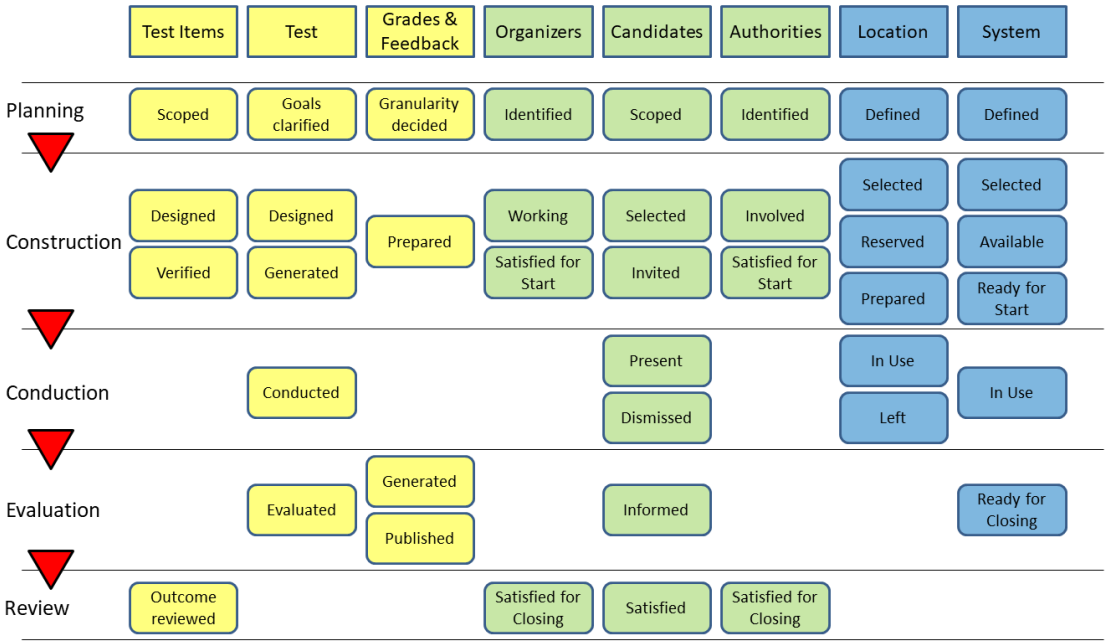


Figure 4.2.: Overview on the assessment process for a summative e-assessment using five phases. The process assumes the application of asynchronous grading, so evaluation happens in a separate phase after conduction.

feedback, organizers and authorities are not expected to change their state during the conduction of the assessment. Also similar to case study 1, alpha "Location" already reaches its final state, as the location is not supposed to be involved in asynchronous grading or review. Consequently, the evaluation phase also has contributions from just four alphas. Three of them also reach their final state in this phase. One of them is the e-assessment system, as we assume that it is not needed for the review of results. For scenarios in which this assumption is not true, the final state can be shifted to the review phase. The same is true for the state "Informed" for alpha "Candidates". In this case study we assume that dissemination of evaluation results is part of the evaluation phase, but it can also be declared to be part of the review phase. Otherwise, the review phase is the same as in case study 1.

Notably, we can skip the alpha "System" from the process and retain a process that represents a traditional written exam which is graded manually after conduction. This stresses the point that integration of e-assessment is primarily a matter of introducing additional steps or stakeholders into a process, but does not require to change existing processes in general. Moreover, this observation proves the point that it can be meaningful to design e-assessment systems by trying to replace traditional manual tasks by automated actions one-to-one.

Work products that are potentially involved in this process include lists of candidates, test item sets, tests, submissions and feedback reports. This is a considerably larger

amount of work products than in case study 1. Most of them will exist only electronically, but are nevertheless considered to be work products. Test items and tests will be created by activities that contribute to the activity space for creating the assessment, while the creation of submissions contributes to the space for conducting the assessment. Feedback reports are created by activities contributing to the space for post-processing the assessment. The activity space for coordinating people will contain activities for communication with authorities as well as with system administrators, as an e-assessment system is involved in this scenario. For the same reason, the activity spaces in the logistics area of concern will contain various activities related to the e-assessment system as well. This is also a considerable difference to case study 1, in which only few activities were expected in this are of concern.

4.3. Case 3: A Distributed Formative E-Assessment

The third case study considers a less formal setting than the others before by looking at an e-assessment where participants can work from at home using a web-based system. This scenario applies to many kinds of homework exercises, but not to formal exams that can be taken from at home due to special situations like a pandemic. The latter is already covered by the previous case studies, as the alpha "location" makes no assumption that participants and assessors or proctors actually sit in the same physical location. Hence, the main difference in this case study is not the location but the assumption that candidates receive immediate feedback from the system after they have made a submission and can improve their previous answers or proceed with subsequent tasks based in their answers so far. We also assume that the content of the exercises is to some extent generated dynamically, e. g. by randomization of variables or by reacting to input in previous steps of an exercise.

Again we can identify specific characteristics of this scenario with respect to kernel and phases: As this scenario does not represent a formal exam, we do not need to include the alpha "Authorities" in our process description. As we use direct feedback and possibly questions that are generated instantly based in previous submissions, it is suitable to join conduction and evaluation phase again as we already did for the oral exam (case 1 above). Notably, we do not need to pay special attention to alpha "Location", although the scenario does not define a single physical location in which the candidates will meet. Instead, we make use of the fact that "Location" is defined abstract enough to represent a physically distributed location which virtually consists of the private workplaces from where the candidates take part in the assessment. The resulting process description in terms of alpha states assigned to phases is depicted in figure 4.3.

Again the planning phase is the same as it was in the previous cases. The construction phase also looks very similar, but shows two differences: State "Generated" for alpha "Test" has been moved to the next phase, while it was member of the construction phase in case study 2. As already mentioned there and discussed above, this reflects the fact that contents of the test are generated individually for each participant. Notably, that does not mean that the test items change their state during conduction of the assessment.

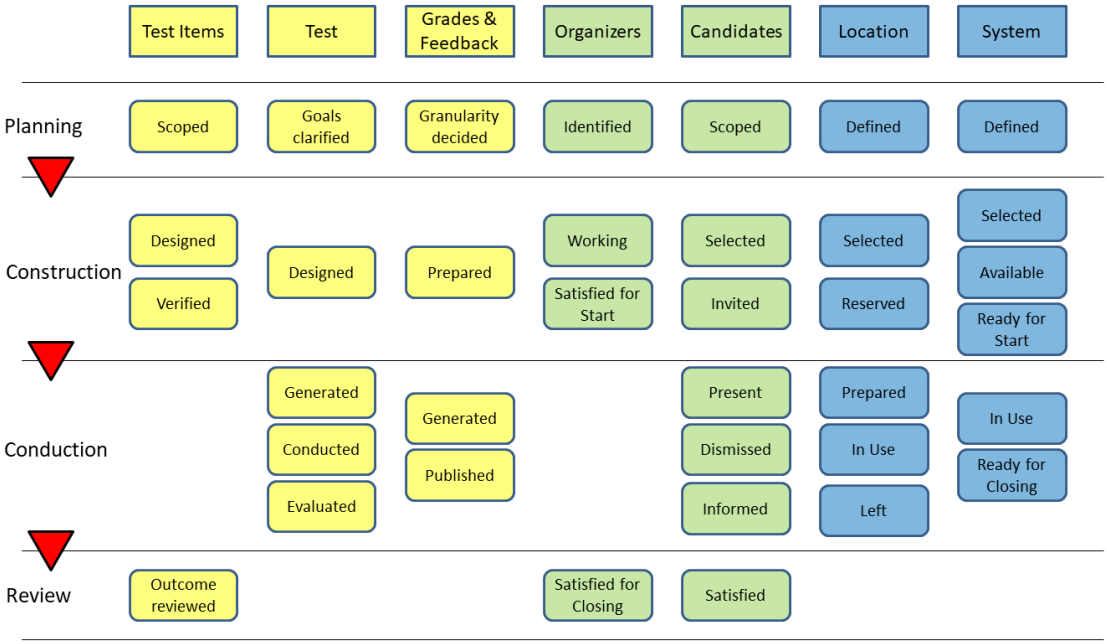


Figure 4.3.: Overview on the assessment process for a distributed formative a-assessment using four phases. The formative setting allows to skip the alpha "Authorities" from the process description. Alpha "Location" is included, although candidates are not required to show up at the same physical location.

The second difference concerns state "Prepared" of alpha "Location". The fact that this scenario considers a physically distributed location is the reason for placing this state in the conduction phase, whereas it was placed in the construction phase for the oral exam (case 1 above). In a distributed formative assessment, there is no possibility to ensure that all candidates have completed to set up their personal workspace before the assessment starts. Hence individual workspace preparations may happen while other candidates are already submitting solutions or even have finished the test. This is in particular true for assessments in which there is some time frame in which the individual assessment can be started.

Besides this change in the "Location" alpha, the arrangement of the non-optional alphas that are present in both in this process description and the one for the traditional oral exam (case 1 above) is exactly the same. Hence from a process point of view there seems to be little difference whether there is a human assessor who iteratively asks questions and evaluates answers or whether this is done by an e-assessment system.

Work products considered in this scenario are almost the same as in the previous case study. Consequently, also the same activities can be expected to fill the activity spaces, but without the need for communication with authorities. Another difference can be found when asking who has to perform some of the activities. In a formal setting, organizers will most probably be responsible for preparing the assessment location and

hence perform the related activities in the logistics area of concern. However, in this less formal setting, candidates can work from at home and hence they are also responsible for preparing their work place. Consequently, the activities may stay the same, but they are performed by candidates instead of organizers.

4.4. Case 4: A Lightweight Ad-hoc Assessment

So far, all case studies only differed slightly in the number of alphas and phases they include and in the arrangement of some of the states into phases. Hence even the less formal setting in case study 3 created a considerably large process description. However, the kernel and the phase model can also be used to represent much more lightweight processes by skipping not only alphas, but also some states of alphas. Thus this case study considers a scenario in which an assessor interacts spontaneously with some candidates just where they are. It is very unlikely that a process description for this scenario will be used to guide the assessor in this process, but it can be used descriptively to explain what is going on.

The process differs in several points from the ones discussed so far: First, we can exclude alphas "Authorities", "System" and also "Location", as the assessment is informal, includes no e-assessment system and can happen anywhere. Second, we can exclude several states of some of the involved alphas: As the assessor interacts with the candidates who are just present, we can exclude the first two stages of alpha "Candidates". Thus "Present" is the first state for candidates to be considered in this process. With similar arguments, we can also exclude state "Identified" for alpha "Organizers". Third, the scenario poses less strict requirements with respect to verification and review of assessment contents. Hence we can exclude the last two states for alpha "Test Items" as well as the final state for "Organizers". The resulting process description in terms of alpha states assigned to phases is depicted in figure 4.4.

The remaining states of the five alphas can then be grouped into just two phases. The construction phase consequently contains the first two states for "Test Items", "Test", "Grades & Feedback" and "Organizers". It thus describes the time frame in which the organizer thinks about doing the assessment and plans what to ask. As we assume this scenario to be a spontaneous assessment, no preparations have happened before. Candidates are not involved in this phase. The other phase is the conduction phase in which only "Test", "Grades & Feedback" and "Candidates" are involved in terms of changing states. This phase is rather similar to the ones in the other case studies besides the fact that state "Satisfied" for alpha "Candidates" is also included here. The idea is that in an ad-hoc assessment any appeals are handled directly (possibly by asking just another question if a candidate is dissatisfied with the previous one) and thus no formal review phase is needed. As already discussed above, the organizer is also not interested in detailed verification and review and thus the respective states from the review phases in the other case studies are simply skipped here.

One could think of making the process description even smaller by skipping state "Dismissed" for alpha "Candidates". This would stress the point that the assessment can

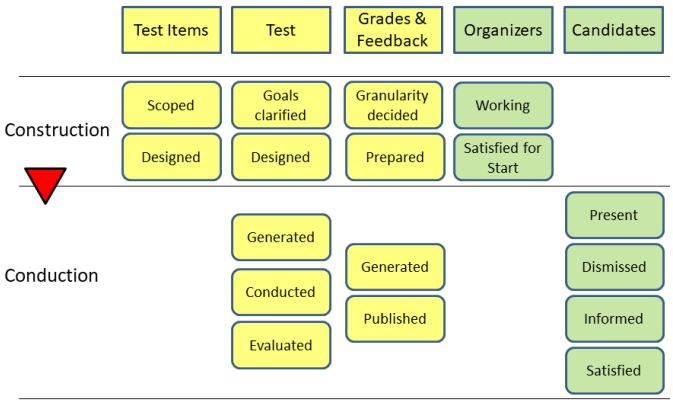


Figure 4.4.: Overview on the assessment process for a lightweight ad-hoc assessment using just two phases. The very informal setting allows to skip the alphas "Authorities" and "Location" from the process description. Also "System" can be skipped as this assessment is not considered to be an e-assessment.

happen anywhere and candidates are not required to come to a certain location (and consequently leave it later). On the other hand, one can understand the state "Dismissed" also in a less literal way and consider a candidate dismissed once the organizer stopped asking questions to the candidate. Notably, the ESSENCE standard allows to make customizations to states in terms of adding or removing checklist items. Consequently, one could define a even more fine-grained adoption of the kernel for this particular scenario by changing the checklists but keeping the overall idea of each of the alpha states included in the process description.

Similar to skipping alphas and states, this scenario can also skip several work products and activities that were used in the other case studies. If the assessment is conducted orally and results are not reported in any way, there may be no formal work products in this process at all. There may also be no need to perform any activities in the activity space for identifying stakeholders or coordinating people. Finally, as no alpha from the logistics area of concern is involved in this process, also all activity spaces for that are can be ignored.

5. Results

The previous chapters were guided by the question on what changes to traditional assessment processes are necessary if e-assessment options are integrated. On the way towards an answer for that question, we defined a notation for assessment process models and used it in some exemplary case studies. These efforts now allow us to answer the original question, characterize integrated e-assessment from a process point of view and also discuss some findings beyond the scope of this publication.

5.1. Contributions to Integrated E-Assessment

First of all, the universal kernel made one explicit reference to e-assessment by providing an alpha dedicated specifically to e-assessment systems. Moreover, the concept of ESSENCE allows to add more specific aspects like tracking states of a system administrator as one of the organizers by adding a sub-alpha. These observations can be interpret in two ways: First, we can conclude that electronic assessments add some amount of complexity to the general concept of assessments by introducing new entities to interact with and new duties to be picked up. Second, we can conclude that integrating these new elements into the existing collection of concepts, entities and duties related to assessments is possible without problems. In particular, the new entities can be added as alphas or sub-alphas as needed and thus integrate smoothly into existing structures by extending them but not breaking them.

This smooth integration does not only apply to the kernel as such, but also to the process descriptions that are based on this kernel. In particular, the comparison of case study 1 and 3 showed that there is not necessarily much difference between a traditional oral exam and an e-assessment besides the fact that a alpha “System” was included in one case but omitted in the other. Hence from a process point of view there is indeed no need for major changes to existing processes when switching from non-electronic assessments to e-assessments. Instead, the e-assessment system can be added and pick up some of the duties that were performed manually earlier. At the same time, adding the additional alpha to the process also adds additional activities related to e-assessment systems, so that processes involving e-assessment tend to be larger in terms of the number of activities than processes for non-electronic assessments. However, this does not necessarily mean that the processes get more complex or more expensive in terms of time consumed by the activities, as automated activities may also help to save time or reduce the need for communication.

Most of the activities added by introducing e-assessments related to the logistics area of concern, which is not surprising as the respective alpha is also located there. The people area of concern was less affected, which can be interpret the following way: Using

electronic assessment systems does not primarily change what to do (besides adding some activities as mentioned above), but the way how it is done. Putting it this way, e-assessment systems are hence tools that are integrated into the assessment process and allow to do certain things in a certain way, just as any other tool does as well. This implies a strong demand towards the development of e-assessment systems that these should produce as few limitations to the assessment process as possible, as one would expect a tool to be helpful but not limiting.

Besides adding alphas, sub-alphas or activities to kernel and processes, some checkpoints also change their characteristics if e-assessment systems are used. Hence we can conclude that there are not only explicit relationships, but also some implicit ones. The amount of those depends on the features of the selected e-assessment system used in an actual process, as not all e-assessment systems can perform the same activities. This is particularly true for the aspect of grading the assessment. Hence the question of integrating e-assessment does not only influence processes by using an e-assessment system at all, but may also influence processes when changing from one system to another.

The phase model used for process descriptions deliberately made no strict assumptions about sequencing activities. Also activity spaces do only provide a very course-grained view on the order in which activities will be performed. However, when it comes to enacting processes in an actual institution, the order of activities can have some important effects. The most important question in this context is the question on whether to choose test item types first or to choose the system first. From a didactic point of view the preference would be to first scope the assessment and decide on appropriate test item types and then select an appropriate e-assessment system to be used. However, in practice there is often a very limited set of e-assessment systems available at a particular institution and organizers have to decide whether these offer a sufficient selection of test item types for the purpose of the assessment. Consequently, integration of e-assessment happens by adapting the form of assessment to the available systems instead of adapting the choice of systems to the desired form of assessments. As changing the form of assessment is neither desirable nor possible in some cases, we can conclude that extending the features of e-assessment systems in terms of available test item types or feedback mechanisms is an important driver for the successful adoption and integration of e-assessment.

A similar problem is the question on when to choose the location of the assessment. In the same way in which the choice of the system may limit the choice of test item types, the choice of location may limit the choice of available systems or vice versa. Hence another driver for the successful adoption and integration of e-assessment is to develop e-assessment systems that can be used as independent from locations as possible. That challenge became particularly urgent in the time of a global pandemic when it was not possible to conduct assessments in specific locations that were prepared for that purpose. Nevertheless, the choice of location may also be constrained by other factors such as the availability of additional lab equipment, which in turn may result in some implications on the available computer infrastructure.

In summary, we can state that the use of e-assessment systems can be integrated seamlessly into existing assessment processes in general and that the features offered by a

particular e-assessment system are the key factors that may ease or limit the integration in a particular process or institution.

5.2. Contributions beyond the Scope of this Publication

As the results so far show that e-assessment requires neither entirely different processes nor major changes to existing processes, we can conclude that the kernel for educational assessment is indeed universal. Hence it contributes to a much wider discussion as it helps to unify the terminology and to structure and compare scientific contributions to different aspects of assessments. Using the kernel and the phase model, any scientific contribution can clearly state which aspect of the educational assessment process it addresses in terms of alphas, alpha states or phases. One could also use the kernel as basis for a systematic literature review to find out whether there are alpha states that are only rarely covered by research and that are hence potentially less well understood in the scientific discussion.

Moreover, practices on how to plan, prepare, conduct and evaluate assessments can now be described in a structured but still flexible way in order to be evaluated and compared. This can ease research as well and also help assessment organizers in finding ways to share knowledge and improve their assessments. This particular contribution is strengthened by the fact that tool support for process definitions is available.

Despite these achievements, this publication did not dig deeper into the business of defining and analyzing assessment processes. The case studies discussed in chapter 4 are thus only starting points that can be extended and refined in order to create full process descriptions that represent the specific habits of a particular institution. There are also a lot more than the four different kinds of assessments that can be described and among these there are possibly many interesting corner cases that require very sophisticated modeling. This opens the door for the definition of domain-specific alphas that may extend the kernel and help to cover more details. However, it can be assumed that the general way of modeling assessment processes based on a kernel is stable enough to realize all these extensions without the need for rethinking assessment process descriptions in a fundamental way.

Part II.

Engineering E-Assessment Systems

6. Design of Technology-Enhanced Learning Systems

Software designers have many choices where and how to implement e-assessment features. There are systems existing specifically just for the purpose of e-assessment, there are e-learning systems offering some e-assessment features explicitly, there are e-assessment components that can be embedded into other systems, there are systems that provide some e-assessment features implicitly, and there may be even more mixtures of these and other ways of implementing e-assessment features.

This wide range exists due to the fact that technology-enhanced learning systems can have different scopes and duties. For instance, a general purpose learning management system offers features for the handling of courses, the dissemination of learning materials, the discussion among students and possibly also for training and exercise. In this case, e-assessment features are optional, as a learning management system can also be used in a meaningful way without having these features. On the other hand, an intelligent tutoring system must have some implicit e-assessment features as it would otherwise not be able to gather information about a student's capabilities and thus would also not be able to give meaningful tutoring advice. Finally, in an exam students or teachers may wish to use a system which focuses solely on the purpose of the assessment and does not disturb them with additional features.

On a general level, this wide range also exists for other classes of systems. For example, computer games are typically known for their emphasis on high-quality graphics and user experience, but often also include features from social media or e-commerce platforms. Consequently, a strict distinction whether an online platform is a social media platform also offering games or a games platform also offering social media services is not always possible. Hence, having a closer look on separate components offering specific features and also looking on possible ways to integrate these with each other seems to be a suitable way to look at systems not only in the domain of education.

This part of the publication first examines the specific features of technology-enhanced learning systems in more detail in the following sections. These sections also include several topics related to software architecture for e-learning and e-assessment systems as found in the literature and technical standards. The purpose of these sections is to elicit a list of typical building blocks of e-assessment systems. Based on that, chapter 7 provides several static and dynamic views on e-assessment systems, that can be used as a conceptual framework in terms of architectural patterns while discussing seamless technical integration of e-assessment. The use of this framework is demonstrated in several case studies in chapter 8. Conclusions with respect to the essential qualities

e-assessment systems must ensure by design and aspects that need further research both in integrated e-assessment and beyond are summarized in chapter 9.

Besides the different scope and duties of technology-enhanced learning systems also the domains they are used in may influence system's design. The more domain-specific a particular e-assessment feature is, the less useful will it be to include it into a domain-independent system. From a software engineering point of view this can be solved by embedding domain-specific components into more general architectures (e. g. by using service-oriented architectures as suggested in [5]) or by designing domain-specific system variants. Both ideas will be touched throughout this chapter from a software engineering point of view, while the discussion of actual domain-specific requirements is deferred to part III of this publication.

6.1. System Components

Following a general trend in system design and system architectures in recent decades, technology-enhanced learning systems transformed in three generations from monolithic blocks via modular systems to service oriented frameworks [61]. This is a comprehensible development due to the many similarities between technology-enhanced learning systems and other software products. Consequently, there is also a trend in very recent years to move forward to cloud based solutions in e-learning and e-assessment, which is considered a fourth generation by some authors [115]. These trends were not only driven by purely technical innovations, but also by actual requirements in the context of these systems. Service oriented architectures were in particular introduced due to the need for sharing materials or assessments across courses and teachers or even institutions [63, 18]. A similar need for sharing expert systems and knowledge modules also led to modularization in the area of Intelligent Tutoring Systems (ITS) [80], which usually also include some kind of assessment features. Learning management systems (LMS) also included a rising amount of e-assessment features. Especially those systems that are developed (as open source projects) by a distributed community (such as Moodle or Ilias) benefit from modularization. With rising numbers of students and in particular rising numbers of electronic assessments, scalability became an crucial issue for e-assessment systems in particular and thus put arguments in favour of cloud solutions to the front [231].

While the notion of different generations of systems according to their architecture refers to the internal structure of these systems in the first place, modularization also is a necessary prerequisite for constructing integrated systems. Integrated e-assessment as the main theme of this publication is not possible from the software engineering perspective without understanding technology-enhanced learning systems as a composition of components and services. Although situations might exist in which a system offering only e-assessment features is appropriate to use, ITS or LMS can be expected to integrate e-assessment capabilities either as own components or as external services. As a consequence, there will be no strict definition on how to tell an LMS with e-assessment features from an e-assessment system with LMS features and alike. The following subsections hence report on different kinds of components found in the literature, that typically appear in the

context of technology-enhanced learning systems and that may be integrated with other components in a system offering e-assessment features. The goal of this section is hence to compile an overview including a rough description of component interfaces as a baseline for subsequent considerations on architectural patterns in the next chapter. The literature study particularly included (amongst other sources) a systematic review of papers from the *International Conference on Technology Enhanced Assessment* (TEA) (formerly known as *International Conference on Computer Assisted Assessment* (CAA)), the *IEEE Global Engineering Education Conference* (EDUCON), the *International Conference on Intelligent Tutoring Systems* (ITS), the *IEEE Transactions on Learning Technology* (TLT) and the *Special Issue on eLearning Software Architectures* issued by *Science of Computer Programming*.

6.1.1. Introductory Remarks on Component Design

There are at least two fundamentally different ways on how to split a system into components, which may depend on general design decisions as well as technical constraints such as the programming language used. A simple example is the architecture of the Learning Management System Moodle: The system is implemented in PHP and structured in a core, modules and plug-ins. Within this architecture, each module or plug-in realizes the full stack from user interface to database queries as far as necessary [46]. Hence the system is divided primarily into vertical slices, where each slice is a larger component that may be subdivided into small components by horizontal slices to make some bits reusable. This is somewhat similar to the ECMA "Toaster" model [78].

Different to that, a classical three-tier architecture as used in many information systems divides a system primarily into horizontal slices (often called layers). Each slice has an abstract duty such as user interface, business logic or data storage. Typically, each slice is subdivided into several components realizing or contributing to different features of the system [43]. The organization of presentation in the following subsections is biased towards the latter style for practical reasons, but does not exclude components from systems following the former style explicitly. A more detailed discussion of these two ways of component design is deferred to section 7.1.1.

A specific security problem in e-assessment that is not relevant in all domains is the execution of student source code as it happens in the automated assessment of programming exercises. Malicious code can be submitted by student intentionally to attack the grading system, but also unintentionally by flaws or naive experiments. A typical general solution to this problem is to design a separate system component for grading untrusted code and to apply sandbox mechanisms and tools to that component [177, 182]. A particular technical solution in this context is to use Docker as a universal technological framework in this approach [318]. These specific run-time environments that may be beneficial when including certain components in a system are not in the focus of the following subsections and are thus only discussed where appropriate.

6.1.2. Overview

During the literature review, a total amount of 38 publications from conferences or journals has been identified as relevant. This does not imply that only 38 technology-enhanced learning systems exist that are relevant in the context of the research. Instead, it means that only these publications included sufficient information on the system architecture to be useful in a literature review on system components. 23 publications also included some sort of diagram of the architecture, which was considered helpful but not mandatory for an analysis of the presented architecture. There is also a high probability that there are more publications available from sources that have not been scanned for this literature review. While each of them will add an additional data point to the findings, not having included them does not diminish the value of the results from the literature review. However, all information on quantity of publications given in the following subsections should be understood as rough estimates and not as an empirical classification.

Components found during the literature review have been grouped in four categories, where each category contains three to five components, resulting in a total set of 15 components. As there is no common terminology used throughout the publications, there is also a list of synonyms for most of the components. Each category of components is discussed in one of the following sections. Each section contains a short introduction to the category, a description of the contained components and a table listing the components, their definitions and the references in literature. A summary table is presented in the summary section after discussing all categories.

6.1.3. User Interface Components

As technology-enhanced learning systems need to receive input from users, user interfaces of various kinds occurred in the literature review and naturally formed a category, which has also been identified earlier by other authors (e. g. [73]). In the literature review, three main user interface components could be identified, where one of them faces the students and two face the educators or administrators. An overview on these components is given in table 6.1.

A *student frontend* (also called student LMS, student VLE, student CMS, student agent, or learning interface) offers features to display assessments to the students and to retrieve their answers. The student frontend is thus typically highly interactive and the amount of different item types supported by an e-assessment system is typically determined by the amount of different types of interactions the student frontend is able to offer. This in turn explains the large amount of papers on student interfaces, as publishing new features in this area appears highly attractive for the community. Systems often provide one single student frontend component, which is extensible by plug-ins (see section 7.3.1). However, there may also be cases in which a system offers more than one student frontend component, such as one browser-based frontend for general purpose and one app-based frontend specifically designed for mobile devices.

Component Name and Synonyms	Features / Functionality	References in Literature
Student frontend (also: student LMS, student VLE, student CMS, student agent, or learning interface)	get and display assessments, retrieve and store answers	[11] [18] [39] [48] [58] [80] [106] [108] [115] [116] [121] [140] [156] [183] [184] [198] [228] [229] [253] [306]
Teacher frontend (also: teacher LMS, teacher VLE, teacher CMS, or admin agent)	administration, authentication, assessment scheduling	[11] [115] [168]
Authoring tool (also: item bank user interface)	create contents	[5] [58] [63] [116] [168] [181] [184] [198] [228] [253] [187]

Table 6.1.: Different types of user interface components for e-learning and e-assessment systems found in literature.

Notably, the student frontend is not limited to inputs via keyboard and mouse, but may also be able to process speech input. It may also include sub-components that create reactions on that speech input on the client side independent of the further processing of input within the underlying technology-enhanced learning system [144]. More details on the different ways of distributing functions between client and server are discussed in section 7.4 in the next chapter.

A *teacher frontend* (also called teacher LMS, teacher VLE, teacher CMS, or admin agent) offers features for administration, authentication, and assessment scheduling. It thus aggregates the features related to the organizational aspects of assessments. As these are in the focus of research more rarely, publication counts for these interfaces are lower than for student interfaces, which does not imply that these interfaces are offered more rarely by e-assessment systems.

More often, an *authoring tool* is discussed explicitly in literature. It offers features required to create contents, which in particular refers to assessment items, item pools, and grading schemas. It thus aggregates the features related to the educational aspects of assessment and is related more closely to the student interface and its features. Thus it is more in the interest of research and thus mentioned more often in literature, but also remarkably often by commercial tool. Notably, the naming difference between between the teacher *frontend* and the authoring *tool* is intentionally. While the former is often designed as a closely coupled component within an technology-enhanced learning system, the latter is often designed and perceived as a standalone tool. Nevertheless it can also be

Component Name and Synonyms	Features / Functionality	References in Literature
Assessment generator (also: instructional manager, curriculum agent, task selector, tutoring component, or steering component)	create assessments from item pool, individualize training	[18] [48] [73] [80] [106] [108] [115] [121] [155] [169] [181] [183] [201] [229] [306] [315]
Item generator (also: problem generator, item constructor, exercise generator)	generate items/problems	[18] [39] [97] [106] [181] [184] [229] [187] [315]
Pedagogical module (also: hint generator, tutoring engine)	provides advice like teachers or hints	[73] [72] [80] [106] [108] [121] [183] [184] [198] [229] [306]
Evaluator component (also: backend, checker, diagnose module, assessor, grader, marks calculator)	analyse submissions and mistakes, create feedback	[5] [11] [18] [39] [48] [58] [79] [97] [106] [108] [116] [121] [127] [140] [155] [169] [183] [184] [198] [228] [229] [253] [306] [187] [315]
Domain-specific Expert System (also: problem solver, domain component, knowledge agent)	perform domain-specific operations or analyses	[5] [39] [58] [73] [106] [181] [187]

Table 6.2.: Different types of educational components for e-learning and e-assessment systems found in literature.

understood as a loosely coupled component within a (distributed) technology-enhanced learning system.

6.1.4. Educational Components

The core of e-assessment systems are their educational qualities and thus the algorithmic power they offer for generating contents, providing advice, and evaluating answers. Components that are concerned with these features are grouped in this category. The literature study identified five components that relate to this area. An overview is provided in table 6.2. They are discussed here in the order of appearance during an assessment.

An *assessment generator* is concerned with preparing an assessment for delivery to the student. This often includes selecting appropriate items from an item pool in case of adaptive system behaviour in order to individualize training or assessment, but can also appear in non-adaptive context in which nevertheless a particular exam needs to be retrieved from a database to be delivered to a student. As the former case attracts a lot of research, it is highly present in the literature.

An additional *item generator* is concerned with filling item templates with actual content, for example by creating random numbers. Consequently, it is not used in context in which fixed items are used and in which any adaptations are performed by the assessment generator mentioned above. This explains the lower number of occurrences in the literature. Different patterns on how the item generation process is organized are discussed in the next chapter in section 7.4.2.

A *pedagogical module* is concerned with providing advice as a human teacher or tutor is expected to do based on a didactical analysis of an actual situation. A typical action triggered by a pedagogical module is to provide hints to students while they work on an assessment item. Consequently, these components primarily occur in assessments that focus on learning, training, or tutoring instead of formal evaluation of student performance. Notably, a literature review from 2009 [210] explicitly makes a distinction between plain feedback on correctness (which would refer to a evaluator component discussed in the next paragraph) and more intelligent analysis as required by a pedagogical module. Although one would expect the latter to be a crucial part of intelligent tutoring systems, that literature review reports a low occurrence rate of components for intelligent analysis of student solutions in intelligent tutoring systems (3 out of 34).

An *evaluator component* is concerned with analyzing submissions from students and identifying mistakes that may occur in these submissions. As part of that, it is also concerned with the generation of feedback that is presented to the student. It is hence somewhat similar to the pedagogical module mentioned above (and may be used by these modules), but it may be much simpler in that it basically just applies a grading schema to a solution but is not able to provide any hint on how to improve a wrong solution. As this seems to be sufficient in several situations, an evaluator component is mentioned much more often in the literature than a pedagogical module.

A *domain-specific expert system* is an external component that is not specific for the purpose of e-learning or e-assessment, but is able to solve general problems in a particular domain. Computer-algebra-systems (CAS) are typical examples of this type of components. Domain-specific expert systems may be connected to evaluator components to enable complex analyses or to item generators to allow for sophisticated generation mechanisms. Notably, some system architectures include components with "expert" in their names, but not in the meaning of domain-specific expert systems. Instead, they refer to expert capabilities in terms of pedagogical interventions that are covered by pedagogical module as defined above or grading capabilities that are covered by evaluator components. Similarly, [72] include a "problem solver" in their architecture, but use it for "inferencing, case-based reasoning, student model evaluation, and other tasks normally associated with learning and teaching activities", which is considered to be the duty of a pedagogical module in this publication.

Component Name and Synonyms	Contents and structure	References in Literature
Item bank (also: Question Bank, Repository of Questions, Exercise Database)	assessment items including rules on how to grade responses and generate feedback or hints	[63] [108] [121] [140] [155] [169] [181] [229] [306]
Domain knowledge model (also: Knowledge base)	information on the domain of the assessment such as facts and concepts, organized by relations or rules	[58] [73] [72] [80] [106] [183] [198] [229]
Student model (also: Learner model)	information on a particular student such as competency levels or overall scores, organized as records referring to an underlying competency structure	[48] [58] [73] [72] [80] [106] [121] [140] [155] [183] [198] [201] [229] [306]

Table 6.3.: Different types of knowledge representation and storing components for e-learning and e-assessment systems found in literature.

6.1.5. Knowledge Representation and Storing Components

Virtually any e-assessment system contains a component for general data storage for users, assessment items, and solutions. These very basic features are common to almost every information processing systems and are thus largely out of scope for this literature study. However, there are also components for storing more specific data, which are often mentioned in the context of intelligent tutoring systems or adaptive assessment systems. Thus an extra category for data-storage components is helpful. An overview of the components contained in this category is provided in table 6.3.

An *item bank* stores assessment items including rules on how to grade responses and generate feedback or hints. Authoring tools are typically the only components that have write access to an item bank, while problem generators and assessment generators may have read access. The internal structure of an item bank may vary a lot, ranging from a simple list of items to sophisticated catalogues that are searchable via elaborated meta data.

A *domain knowledge model* is responsible for storing information on the domain of the assessment that is not specific to a certain assessment item, but reflects facts or competencies of the particular domain. Domain knowledge models are mentioned most often in conjunction with expert modules that are able to evaluate a submission by using domain knowledge, but without knowing the correct answer to the particular assessment item explicitly. The same goes for connections to pedagogical modules that use domain knowledge to generate hints.

A *student model* is responsible for storing information on a particular student that again is not specific to a particular assessment item. Instead, a student model reflects competencies or similar properties that relate to the person and his or her capabilities or performance. Student models are mentioned most often in conjunction with adaptive system behaviour, where adaptivity is based on the information stored in the student model.

Additional domain-specific data storage is mentioned only rarely in the literature [156, 127]. It is relevant only in domains in which submissions to assessment items are large or complex objects, such as program code in the domain of programming assessment. Consequently, specific components for this purpose are explored only in conjunction with these domains and almost never as part of general assessment systems.

Specifically for the domain of intelligent tutoring systems it is common to use three models: domain knowledge model, student model and tutoring model (see e. g. [64]). The latter may contain information about different pedagogical approaches or similar. As this is not discussed outside the community of intelligent tutoring system, this specific kind of data storage is not included in the survey explicitly. However, it can be considered as part of a pedagogical module as mentioned in section 6.1.4.

A special case of a data store that is not focused on a particular kind of data and not limited to a single technology-enhanced learning system is introduced in [119]. It combines a composite data structure with tool specific adapters to synchronize data stores between different systems in real-time. Hence it can be considered an additional component explicitly designed for the purpose of integrating different technology-enhanced learning systems. Although not explicitly mentioned in the original publication, this can also include e-assessment systems.

6.1.6. Management Components

The core features and requirements of e-assessment systems motivate the categories and components discussed so far. However, some components are introduced due to additional requirements or for the sake of better software architectures. They are collected in this category. An overview on these kinds of components is given in table 6.4. In general, these components are far less present in the literature.

A *reservation service* realises an additional feature of e-assessment systems reported sometimes in the literature and by commercial tools. It is responsible for registering students for assessments and thus covers an additional part of the organizational process around assessments that is not necessarily covered by the teacher frontend discussed in section 6.1.3 above.

A *queue* is explicitly mentioned in discussions of system architectures only. It can occur in several directions: (1) It may forward data from some frontend or steering components to evaluator components that may run in parallel on separate systems for performance or security reasons. (2) It may forward data from evaluator components to frontends. While this component does not add any particular feature or educational value to a system, it may be crucial for several architectural patterns on how to connect other components. Details on that will be discussed in sections 7.1.2 and 7.2.2 in the next chapter.

Component Name and Synonyms	Features / Functionality	References in Literature
Reservation service (also: Scheduling)	Register students for assessments	[5] [156] [168] [181]
Queue (also: Spooler, Middleware, Service Broker)	connects frontend components to evaluator components or the other way round for continuous data transfer	[5] [11] [79] [97] [127] [228]
Data transfer component (also: Notify and announce, Reporting agent, Assessment commit agent)	bulk transfer of data, such as publishing results or archiving assessments	[18] [115]
Infrastructure agent	starting and shutting down instances	[115]

Table 6.4.: Different types of management components for e-learning and e-assessment systems found in literature.

A *data transfer component* is also only mentioned in discussions of system architectures. When used, there may be several components of that kind for a specific purpose rather than one abstract general purpose component. Different to a queue, a data transfer component is not concerned with continuous forwarding of data, but performs bulk transfer of data between components. It may also be introduced to a system for performance or security reasons and is of particular interest in conjunction with distributed data storage (as discussed in section 7.3.2 in the next chapter).

An *infrastructure agent* is reported for cloud-based solutions only. It is responsible for starting and shutting down instances of other components to adjust the size of the running system to the current needs. It is only necessary in systems which are aware of being a cloud system. Different to that, components can also be deployed as services in a cloud based or container based environment in which the underlying cloud or container infrastructure is responsible for starting and shutting down additional instances.

6.1.7. Summary

The findings from the literature review and component classification are summarized in table 6.5. Besides the plain numbers of occurrences, the literature review also showed that there is not always a strict distinction between components. This applies in particular for the category of educational components, where the same functionality can e. g. be understood as part of a pedagogical module or an evaluator component. At the same time it can be stated that the review revealed no additional and fundamentally different components than the ones discussed so far. Thus, it can be concluded that the list is

	User Interface			Educational					Knowledge Repr. / Storing			Management			
Ref.	Student Frontend	Teacher Frontend	Authoring Tool	Assessment Generator	Item Generator	Pedagogical Module	Evaluator Component	Domain-specific Expert System	Item Bank	Domain Knowledge	Student Model	Reservation Service	Queue	Data Transfer Component	Infrastructure Agent
[3]	●	●	●	●			●		●						
[5]			●				●	●				●	●		
[11]	●	●					●						●		
[18]	●			●	●		●							●	
[39]	●				●		●	●							
[48]	●			●			●				●				
[58]	●		●				●	●		●	●				
[63]			●						●						
[72]						●				●	●				
[73]				●		●		●		●	●				
[79]							●						●		
[80]	●			●		●				●	●				
[97]					●		●						●		
[106]	●			●	●	●	●	●		●	●				
[108]	●			●		●	●		●						
[115]	●	●		●										●	●
[116]	●		●				●								
[121]	●			●		●	●		●		●				
[127]							●						●		
[140]	●						●		●		●				
[156]	●											●			
[155]				●			●		●		●				
[168]		●	●									●			
[169]				●			●		●						
[181]			●	●	●			●	●			●			
[182]	●						●						●		
[183]	●			●		●	●			●	●				
[184]	●		●		●	●	●								
[187]			●		●		●								
[198]	●		●			●	●			●	●				
[201]				●							●				
[205]	●					●	●				●				
[212]	●	●					●		●				●		
[228]	●		●				●						●		
[229]	●			●	●	●	●		●	●	●				
[253]	●		●				●								
[306]	●			●		●	●		●		●				
[315]				●	●		●								

Table 6.5.: Summary table for the literature review on system components

exhaustive in the sense that it contains all kinds of components that commonly appear in educational software systems.

6.2. Technical Standards for Technology-Enhanced Learning Systems

Besides system developers and researchers designing e-learning and e-assessment systems and creating components as needed, there are also people working on generalization and unification of such results by standards. Several benefits can potentially be achieved by establishing standards: System designers need not to start from scratch but can make use of a stable base. Interoperability gets more easy by using common interfaces. Comparison of tools also gets more easy by checking conformance to standards. On the downside, standards limit the space for experiments in research prototypes and may thus slow down innovation. There is also a risk of covering not the right range of aspects for a larger audience, potentially resulting in a set of competing standards in the same area. Hence standards were not used as the entry point for the study on system components, but can nevertheless be neglected.

In the area of e-learning and e-assessment systems the most relevant set of standards of issued by the IMS Global Learning Consortium. The following sections thus discuss the IMS Abstract Framework as an umbrella to their various standardization activities as well as the IMS Learning Tool Integration standard and the IMS Question and Test Interoperability standard as these are particularly relevant with respect to e-assessment and system integration.

6.2.1. IMS-AF

The *Abstract Framework* (AF) specification issued by IMS Global Learning Consortium is designed to describe the general context of various other specification documents issued by IMS as well [129]. It is organized in four layers: Application Layer, Application Services Layer, Common Services Layer and Infrastructure Layer. Additionally, it defines a so-called "Sea of Components", which is a catalogue of components that provide services or contain data structures relevant to the domain of e-learning systems. There is no mapping defined in the standard between the services described within the layers and the components defined in the component catalogue. Neither the list of elements for the layers nor the list of elements in the component catalogue is meant to be exhaustive, but to represent a list of typical examples. Although the only available version "1.0 Final" of the standard has a lot of details marked as "to be done", it is worthwhile to look into these list of examples and to compare them to the findings from the literature study discussed in the previous section.

The application layer lists applications that may use the services and components defined within the IMS-AF. In other words, this is a list of applications that can be considered to be part of the domain of e-learning systems. The list includes very general applications such as "Learning Management System", "Student Information System"

and “Portal”, but also more specific ones such as “Assessment System” and “Content Authoring Tool”. The list does no include intelligent tutoring systems as they were considered in the literature study in section 6.1. On the other hand, it also includes applications like “Bulletin Board Tool” and “News Tool” that also occur in domains other then e-learning and that were not covered in the literature study.

The application services layer lists coarse-grained features that can be offered by applications in the e-learning domain. For some of these features, core components are listed. For example, the “Assessment” feature has “Assessment”, “Item”, “Object-bank”, “Section” and “Result Report” as its core components, where all these names refer to elements from the “Sea of Components”. Again the list contains features which are not specific to the domain of e-learning and have not been considered in the literature study, for example “Calendar” and “Group Management”.

The common services layer is organized very similar, but explicitly contains only features that are generic in nature and are thus not specific for the domain of e-learning. This distinction seems to be quite unclear, as the application services layer also contained services that seem to be more generic than specific. However, it is explicitly not the intention of IMS to work on specifications for features included in the common services layer.

The “Sea of Components” is the most interesting part of the specification with respect to this publication. It contains the description of 30 components for each of which IMS intends to develop further specifications. A summary of these components and a mapping to the terms used in section 6.1 of this publication is provided in table A.1 in the appendix. A closer inspection of the component descriptions reveals that most of the components are designed from a data management point of view: The majority of components (11 out of 30) cannot be mapped properly as they are concerned with data details that are out-of-scope for the literature study. Examples include the affiliations or hobbies of the learners. Another set of 9 components can be mapped to the general data storage that was used to summarize the storing of business data that is related very directly to assessments and individual items. Finally, there are 4 components that can be mapped to the student model component, and 2 components that can be mapped to the domain knowledge model. Consequently, there are only 4 components remaining, where one can be identified as a data transfer component and three can be mapped to an assessment generator, evaluator component, or both. Notably, 3 out of 30 components refer to presentation of some content, so they can be mapped to a student interface in addition to the mappings discussed so far.

It can be concluded that the IMS Abstract Framework provides a different point of view on components for technology-enhanced learning systems than the one used in this publication. Nevertheless, the design of the framework with different layers, and components as a main design principle stress the importance of studies on architectural patterns that can be used to create system architectures based on the well-known or even standardized building blocks.

6.2.2. IMS-QTI

The *Question and Test Interoperability* (QTI) specification issued by IMS Global Learning Consortium is designed to define a data model for test items, tests and test results [131]. The goal of the standard is to enable data exchange between various tools and systems involved in the process of creating, conducting and evaluating assessment. The standard has evolved through various versions, where version 2.2.1 from 2016 is the current one at the time of writing this publication.

During the literature study of part I in section 2.1.3 it was already mentioned that the QTI standard presents actors and use cases as part of its overview. This also includes a list of tools or systems involved in these use cases, that can be mapped to the ones identified in the literature study as shown in table 6.6. Several observations can be made from this mapping: First, there is a match for all tools and systems named in the standard, which confirms the results from the literature study. Second, the standard is not concerned about the details of assessment system design, as it refers to a assessment delivery systems only as a whole but not as their individual parts (user interface, infrastructure and educational components). Third, the standard seems to assume a certain process model in which item authoring and test construction are kept separate. In addition, this process model considers adaptive system behavior only in a limited way. The specification explicitly states that adaptive test behavior is only supported due to branching and preconditions for test items. Adaptive behaviour based on student models is not supported. However, the standard also defines the notion of an adaptive item, that is a test item that allows for different scoring in subsequent attempts.

Notably, the last system name “assessmentSystem” listed in the standard might be an error and should read “learningSystem” instead, which would better fit the description. In comparison to the IMS Abstract Framework, “assessmentDeliverySystem” is almost the same as component “Assessment” from IMS-AF, but it remains unclear whether the difference in naming is by intention. The “itemBank” is a special case of the “Object-bank” from IMS-AF, which explicitly names items as one of the objects to be grouped in a data-bank.

The “Assessment, Section and Item Information Model” specification of QTI contains a more extensive list of nomenclature and definitions, which partly repeats entries from the overview. In particular, it also names an Authoring System as well as an Assessment Delivery System. In addition to that, it also introduces Cloning Engine defined as “a system for creating multiple similar items (Item Clones) from an Item Template”. This can be mapped to the problem generator identified in section 6.1. The standard allows this component to be a separate tool with no specific position within the assessment generation process or a part of the delivery system. The “Assessment, Section and Item Information Model” specification also names a Scoring Engine as the “part of the assessment system that handles the scoring based on the Candidate’s responses and the Response Processing rules”. This can be mapped to the evaluator component identified in section 6.1.

However, naming and definition of components is just a prerequisite in the QTI standard for the actual content, which is a data structure definition for different kinds of test

Tool or System Name in IMS-QTI Overview	Tool or System Description in IMS-QTI	Mapping to section 6.1
authoringSystem	A system used by an author for creating or modifying an assessment item.	Authoring tool
itemBank	A system for collecting and managing collections of assessment items.	Item bank
testConstructionTool	A system for assembling tests from individual items.	Assessment generator
assessmentDeliverySystem	A system for managing the delivery of assessments to candidates. The system contains a delivery engine for delivering the items to the candidates and scores the responses automatically (where applicable) or by distributing them to scorers.	Student frontend, Queue, Evaluator component
assessmentSystem	A system that enables or directs learners in learning activities, possibly coordinated with a tutor. For the purposes of this specification a learner exposed to an assessment item as part of an interaction with a learning system (i. e., through formative assessment) is still described as a candidate as no formal distinction between formative and summative assessment is made. A learning system is also considered to contain a delivery engine though the administration and security model is likely to be very different from that employed by a DeliverySystem.	Student frontend

Table 6.6.: Mapping from components listed in the IMS-QTI standard Overview [131] to the terms used in section 6.1.

items and tests. The standard defines a mapping to XML for these data structures and thus implicitly introduces exchange of XML data as a suitable interface for component communication. In particular, this supports the assumption that rendering of test items and tests based on XML data is a specific task for a frontend component, while other components work on test or response data in XML. The details of the data structure are of less relevance here, but will be revisited in part III of this publication.

Another contribution of the standard is the definition of a life cycle for student interaction with a test item and a submission model for student interaction with an assessment. These are relevant to identify components that are closely related to each other by performing work within the same interaction. The life cycle breaks the total amount of interaction between a student and a test item into several "Item Sessions", where each Item Session can contain several "Attempts" and each Attempt can be split into one or more "Candidate Sessions". Candidate Sessions are only relevant to the student frontend component, as they start when an item is displayed to a student and end when a student switches to another items for any reason. The first Attempt is started when the item is presented to a student for the first time. Each time, a student submits a response to the assessment system, this terminates the current Attempt and starts a new one. Submitting a response involves both the student frontend component and an evaluator component in order to create feedback to that submission. Finally, a new Item Session starts each time a student encounters a test item for the first time in a new context, such as in different tests. Item Sessions are mostly relevant to educational components such as assessment generators or problem generators, as they may decide on when to display an item and thus when to start e new Item session or how to generate contents for the item at the beginning of the Item Session.

The submission model may limit the number of possible transitions within the Item Sessions. It provides a mode for simultaneous submission of responses from several items within the same part of an assessment. This implies that students do not receive feedback to individual items until they have submitted their responses to the whole part of the assessment. Typically, this also implies that they are not allowed to make additional attempts.

In conclusion, the brief analysis of the QTI standard proofs that the system components identified during the literature study are sufficient to match and explain both the structural and behavioral aspects of the IMS-QTI specification.

6.2.3. IMS-LTI

The *Learning Tools Interoperability* (LTI) specification issued by IMS Global Learning Consortium is designed to allow for integration of remote tools and content into learning management systems [130]. The specification assumes a web-based environment in which a student interacts with systems via a web browser. The standard has evolved through various versions, where version 1.1.1 from 2012 is the current one at the time of writing this publication. Several learning management systems like Moodle or Ilias implement the Tool Consumer's part of the LTI specification.

The specification is very generic in terms of components that are connected, referring to them just as "Tool Consumer" and "Tool Provider". The Tool Consumer typically would be a learning management system (LMS), while the tool provider can be any system that is supposed to contribute features to the LMS. The specification defines two aspects of communication between both partners: The "basic launch" and the "basic outcome services".

The basic launch defines a model on how to establish a trusted connection between Tool Consumer and Tool Provider which allows the Tool Consumer to redirect users to the Tool Provider without passing sensitive information like log-in credentials. The idea is to provide a link that can be clicked by students within the Tool Consumer's student interface that opens the Tool Provider's student interface in the same browser window or a new one, just as if the Tool Provider would be a component within the Tool Consumer's system. The Tool Provider is informed both about the user and the link that was clicked, as well as possibly additional information associated with the link. The Tool Provider can then offer its features through its own student interface and receive interactions, as long as the student closes the browser window or uses any other appropriate way to navigate back to the Tool Consumer's interface.

The basic outcome services define three services to be implemented by the Tool Consumer that receive plain XML messages in order to receive results, delete results or report the current result. Each result only contains a grade from 0.0 to 1.0 as the only piece of information related to the student's performance. The result also contains a reference to the link the student has used during the basic launch.

While the specification defines a technically solid way of connecting different tools to form a interacting system, the very limited amount of data that can be passed over this connection limits the value of the standard. If the Tool Provider offers a complex feature, such as presenting and grading a complete test, the response send back to the Tool Consumer is still just one single grade between 0.0 and 1.0. Nevertheless the existence of the specification and its quite broad use by learning management systems stresses the need both for a modularized view on components to be integrated into technology-enhanced learning systems and for well-defined architectural patterns to describe the technical way of integration.

A non-standardized alternative has been developed by Fontenla González [109, 302] and offers richer options for information exchange between a learning management system and an external tool. However, this approach assumes that the tool servers offers an appropriate API which just needs to be mapped to a generic tool interface via user-defined binding rules. Consequently, this approach is not suited for general purpose standalone tools, but only for tool servers that are meant to be used in a service oriented way by design.

6.2.4. IEEE LTSA

The *Learning Technology Systems Architecture* (LTSA) issued by IEEE in 2003 specifies a high level system design and a list of components for a wide range of technology-enhanced learning systems [1]. It is designed to provide a framework for describing existing and

future systems and to promote interoperability between systems and components. The standard is no longer maintained by IEEE and thus considered withdrawn. Nevertheless it provides some interesting insights.

The main contribution of the standard is a set of four processes (learner entity, evaluation, coach, delivery) and two data stores (learner records, learning resources) that are interconnected by a large set of different data and control flows. The intention is to define a high-level framework of components and connections, while each concrete system in the educational domain may implement an appropriate subset of this framework. The annex of the standard illustrates the intended usage of the framework by several so-called stakeholder mappings that indicate which components and flows are considered most relevant in a given context. One of these stakeholder mappings is named "assessment-centered" and highlights three processes and one data store. A mapping from these elements to the terms used in section 6.1 is provided in table 6.7 alongside a brief definition or description of each component.

The mapping provides a distinction between LTSA design priorities and additional aspects outside the focus of LTSA. For both categories, it distinguishes between primary and secondary aspects. Evaluation and Learner Records are considered primary aspects within the design priorities of LTSA. This also includes the data flow from the Learner Entity to the Evaluation, the data flow from Evaluation to Coach and the bi-directional data flow between Evaluation and Lerner Records. The standard does not use terms like "user interface" or "frontend" explicitly in this place, but names protocols and formats for the data flow from Learner Entity to Evaluation as part of the concerns. Thus the primary aspects can be mapped without limitation to the components Student frontend, Evaluator component and Student model as identified in section 6.1.

The secondary aspects within the design priorities of LTSA include Learner Entity and Coach as well as bi-directional data flows between these two to exchange learning parameters and also bi-directional data flows from Coach to Learner Records to exchange learner information. Different to what was discussed above, the secondary aspects also include the interface to the learner entity explicitly. This supports the decision to include the mapping to Student frontend. The secondary aspects also explicitly include the functionality of the coach. Due to the different activities that can be performed by the Coach according to its description, it can be mapped both to an Assessment generator and a Pedagogical module.

The standard also mentions two more interesting aspects that are outside the focus of LTSA. The list of primary aspects outside the LTSA focus contains an entry for reporting systems. These were discussed in section 6.1 as one particular kind of data transfer components that can be found in some literature references. By mentioning it here, the standards provides another piece of evidence that these components are a common part of system architectures. The list of secondary aspects outside the LTSA focus refers to adaptive system behaviour. Different to IMS-QTI it refers explicitly to adaptivity based in assessment results and not to predefined branching behaviour within tests or single test items.

In summary, the standard provides a view on assessment systems that handles the core duties of such systems in a very similar way as it is done in this chapter. While the

Component name in IEEE-LTSA	Definition or description in IEEE-LTSA	Mapping to section 6.1
Learner Entity	An abstracted process that represents an abstraction of a human learner. The learner entity may represent a single learner, a group of learners learning individually, a group of learners learning collaboratively, a group of learners learning in different roles, and so on.	—
Coach	A process that performs amongst other the following activities: (1) Negotiates/exchanges learning parameters for optimum learning experience. (2) Receives current assessment information from evaluation. (3) Searches and retrieves learner information relevant to the current learning experience. (4) Searches learning resources via queries for appropriate learning content	Assessment generator, Pedagogical module
Evaluation	An abstracted process that may produce measurement(s) of the learner entity.	Student frontend, Evaluator component
Learner Records	A data store of learner information, such as performance, preference, and other types of information. The learner records may store information (which may be later retrieved) about the past (e. g., historical learner records), but may also hold information about the present (e. g., current assessments for suspending and resuming sessions) and the future (e. g., pedagogy, learner, or employer objectives).	Student model

Table 6.7.: Mapping from components named in the assessment-centered view in appendix D.3.2 of the IEEE LTSA standard [1] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping.

standard covers a much wider range of technology-enhanced learning systems, it does not dig deeper into the details of the particular class of assessment systems and such misses details like problem generators or several kinds of management components.

7. Architectural Patterns for E-Assessment Systems

The previous chapter reported on a large list of properties, common components and standards for technology-enhanced learning systems that have been found in recent literature. Based on these findings, this chapter now connects these bits in order to form patterns that can be considered useful when designing and engineering e-assessment systems. A particular focus of these considerations is on questions regarding integration and thus also on well-defined interfaces that describe suitable connections.

The idea of this chapter is to some extent inspired by the similar idea of architectural patterns for intelligent tutoring systems (ITS) explored by Andreas Harrer et al. 10 to 15 years ago [72, 118]. Unlike in that work, this chapter does not focus on the decomposition of a complete system into parts, but on the discussion of system parts that can be integrated with each other or with other systems in order to create meaningful e-assessment features. To ensure a broader exploration of the design space, it is not limited to patterns found directly in the literature. In more detail, section 7.2 discusses system behaviour and design consequences from integrating different components to realize a single desired behaviour. Section 7.3 discusses patterns for structural aspects of system design, that are not primarily driven by required system behaviour. Section 7.4 finally discusses patterns for functional behaviour of single components that may not necessarily influence general system design.

There is some principle work on general patterns for software architecture [43] as well as on different aspects of system integration in the domain of e-learning [119, 95, 224]. The latter differ from this chapter, as they do not specifically consider e-assessment systems and components. The former provides an important basis on how to describe and catalog architectural patterns, but naturally does not consider the domain of e-learning and e-assessment specifically.

7.1. General Remarks on Architecture Style and Focus

Prior to the discussion of different classes of patterns in the subsequent sections, this section provides an overview on architectural styles and component structures that occur in e-assessment systems. The assumption is, that a system will only use one particular style or component structure (or at least only one particular style for each part of the system, if it consists of several loosely coupled parts), but may use several pattern within the boundaries created by the former decision.

7.1.1. Architectural Styles

As already mentioned earlier in the previous chapter, there are some fundamentally different ways on how to split a system into components [288]. Among those, two are used particularly often on the domain of e-assessment systems: Vertical slicing following a shared memory blackboard style and horizontal slicing into layered virtual machines. However, also peer-to-peer architectures can be considered useful in this domain.

Layered Virtual Machines

According to [288], layered virtual machines can be summarized as a system in which one layer offers services to layers above it.

This principle is often used as a two-tier or three-tier architecture in many types of information systems and can also be found in intelligent tutoring systems (e. g. [265]) and e-assessment systems (e. g. [315]). Each layer has an abstract duty such as user interface, business logic or data storage. Typically, each layer is subdivided into several components realizing or contributing to different features of the system. If components are used that offer only very specific functionality, these are often called microservices [202].

Integrating e-assessment features into an existing system that employs the layered virtual machines style usually means to touch most of the existing layers. Considering a three-tier architecture with separate layers for frontend, business logic and data storage, adding e-assessment features for the first time requires several changes: First, new forms of user input must be added to the frontend layer. Second, some logic for grading submissions must be added to the business logic. Third, assessment items and results must be made available on the data storage layer. However, integrating a new item type that can reuse existing grading mechanisms or integrating a new grading mechanism for an existing item type may only require changes to one of the layers. Finally, it is also possible to realize new functionality by creating new connections between already existing components in different layers.

Shared Memory Blackboard

According to [288], a shared memory blackboard architecture can be summarized as independent programs that access and communicate exclusively through a global repository known as blackboard.

An example using this principle is the architecture of the Learning Management System Moodle: The system is implemented in PHP and structured in a core, modules and plug-ins. The core provides basic functionality to access the database and thus the shared memory. Each module realizes a larger feature such as a forum, lessons or assignments. Plug-ins may add specific features to a module such as a single item type. Within this architecture, each module realizes the full stack from user interface to database queries as far as necessary and is not intended to rely on services from other modules [46]. Hence the system is divided primarily into vertical slices, where each slice is a larger component

that may be subdivided into small components by horizontal slices to make some bits reusable.

As just mentioned in the example, integrating e-assessment features for the first time into a system using a shared memory blackboard architecture means to add a new module. Adding a new grading mechanism or a new item type can possibly be realized by creating a plug-in for an existing module, but it may also require to create a new module that duplicates some of the existing behaviour. The shared memory blackboard architecture is not intended to realize new functionality by creating connections between existing modules.

Peer-to-Peer

According to [288], a peer-to-peer architecture can be summarized as a system in which components hold state and behavior and can act as both clients and servers.

No well-known example of an e-learning or e-assessment system exists that solely uses a peer-to-peer architecture. Nevertheless, parts of such systems can follow this principle when using different agents that are connected to each other. Integrating e-assessment features into such a system for the first time means to add another component to the network of peer-to-peer connection. The same is true for any further extensions, that are also realized by adding new components or at least new peer-to-peer connections.

7.1.2. Architectural Focus

As already discussed at the beginning of section 6.1, there is a general trend in the design of e-assessment systems to move from monolithic systems to modular systems. At the same time, there is a choice on whether a system runs on a server, a client or as an distributed system involving both servers and clients. Notably, a distributed system is necessarily modular, but a modular system is not necessarily distributed.

It seems to be a dedicated design choice on whether one constructs a server-focused system (which may or may not defer some less important UI logic to a client component), a client-focused system (which may or may not use some central server solely for sharing data) or a module-focused system in which the emphasis is on designing well-defined and loosely coupled modules. The TEx-Sys model for building intelligent tutoring systems [265] is an useful example to illustrate these differences in the architectural focus. Three systems were build using this model: The first one is a Windows application and thus a client-centered system that actually does not involve any server component. The second one uses a web-browser as client and thus moves all business logic and data management to a server. The third one offers an own web client application that makes calls to web services located on a server. Thus the first one is purely client-focused, while the second one is clearly server-focused. The third one involves both sides while decoupling the server-side into single web services and can thus be considered module-focused.

This design choice is also relevant for integration, as module-focused development may put an special emphasis on re-usability or integration. For example, Wiris provides no complete system, but components to be integrated into systems to handle mathematical

input an evaluation [187]. Similar, the GRAPPA middleware provides no complete system, but a component for integrating different e-learning frontends with different grading components [97]. On the other hand, there are standards like IMS-LTI that deal with integration of systems on tool level and thus enable integration also for client-focused or server-focused systems that do not provide well-defined integration interfaces on component level.

Notably, some key features of an e-assessment system may also have an impact on the architectural focus. For example, many ITS have an emphasis on using several different agents or expert systems that act in parallel and independent of each other. The same is true for e-assessment systems in the domain of programming that are able to grade assignments in different programming languages. Having many conceptually independent components within one system is a strong argument for a module-focused development. At the same time, some e-assessment systems promote mobile access as one of their most important features. This is an argument for client-focused development, as much effort is needed to support different mobile devices and to make functionality available on the client side even if the connection to a server gets lost.

7.2. Behavioural Patterns

For the purpose of this chapter, behavioural patterns are patterns that define which components are callers and callees or which components are used to (re-)direct calls from one component to another. Furthermore we assume that a particular required system behaviour is the main driver for decisions on behavioural patterns for a given scenario. Note that required system behaviour may also cause the use of functional patterns that have a slightly different focus and that are discussed later on in section 7.4.

7.2.1. General Component Behaviour

As an interactive system, any e-assessment system is supposed to react specifically to a users input instead of delivering the same content all the time for all users. The most obvious example for this behaviour is the grading of a response, which will therefore be discussed in more detail later on in section 7.2.2. However, there are many additional ways in which an e-assessment system can adapt its behaviour to a specific user or the amount of users in general. Components like an assessment generator or problem generator can adjust the difficulty of their output according to the capabilities of a student. A pedagogical module can generate hints based on insufficient tries from a student. An infrastructure agent can start additional instances of other components to adjust system capabilities to a rising number of requests, e. g. during an exam. For these and other cases one can identify two different types of general component behaviour that will be discussed in the following sections. Every system will at least use one of these patterns, but it is also possible that a system uses both for different components.

Passive Services

Passive services are waiting for requests that are directly or indirectly cause by user interactions. They perform some actions upon these requests and then wait for the next request to process. Using this pattern can be considered a standard way of designing business information systems and some literature mentions it as a general principle of system design [11, 61]. The general concept makes no limitation on the granularity of the service or the number of computing resources used. While older realizations tend to define few larger services on a single server, more recent system designs favor small microservices distributed across multiple servers [315, 65].

There are some well-understood benefits in using passive services, i. e. if these services can be implemented in a stateless manner. This allows to handle many user requests with little resources. A specific drawback of passive services in the context of e-assessment is that fact that students may not be aware that they can trigger a particular feature of an e-assessment system. For example, students may not know what to do to request a hint on how to complete an incorrect exercise solution and consequently a passive hint generation service does not do anything. In addition, some other patterns like the *asynchronous pull* pattern from section 7.2.2 below require active components to be functional, so that it is not necessarily possible to construct an e-assessment system using only passive services. On the other hand, at least the user interface of every system can be considered to be a passive service that waits for user request, so that virtually any system will use this pattern.

Active Agents

In contrast to passive services, *active agents* have their own agenda on what to do and thus they perform their actions potentially even without any user input. They can be used both for educational components (such as agents that generate hints or exercise suggestions without explicit request from the user) and management components (such as agents adjusting the cloud infrastructure to the current load). They are particularly common in the domain of intelligent tutoring systems [58, 201]. They overcome the above mentioned drawback that students may not know how or when to trigger a certain feature. On the downside, running active agents requires more system resources than just reacting to user input, which may induce scaling issues on systems used by a large amount of students.

Interoperation between Active and Passive Components

If a system uses both passive services and active agents, there is often a service broker component designed as a queue included in these systems. This queue can be used in both directions: (1) Services can push elements to the queue upon request (e. g. submissions as they are handed in), while agents pop elements from the queue as they like (e. g. a grading agent as soon as it is ready to handle a job). (2) Agents can push to the queue as they like (e. g. results as a grading agent has finished its job), while services pop elements

from the queue upon request (e. g. a student interface when a student wants to review results). More details on this will be discussed in the next section below.

7.2.2. Component Communication for Grading a Response

This section reflects one of the most crucial features of e-assessment systems. A response to an assessment item is entered by a student via some user interface component and then needs to be processed by some evaluator component in order to produce a grade or some other kind of feedback. This connection between the user interface and an evaluator component can be realized in many different ways. All patterns from the following subsections are independent of the number of evaluator components involved. Hence, if two or more different evaluator components are invoked to retrieve two or more separate results for a single submission, each invocation can follow one of the following patterns. Notably, this is only one possible way of parallel grading, while another option is to split the control flow internally within the evaluator. This will be discussed in section 7.2.3 below. While system designers are free to use more than one of the following patterns within one system, they will surely use at least one.

Synchronous Push

One pattern is that of a *synchronous push*, which basically corresponds to a plain method call in many programming languages. In this pattern, user interaction directly triggers the grading process and the user has to wait until the input is processed (see fig. 7.1). Consequently, either a queue or an evaluator component used in this pattern needs to be a passive service that waits for synchronous push requests. Systems in which grading tasks are short running and in which the next step depends on the previous result can employ this pattern. Web-based e-learning systems that are implemented in scripting languages like PHP also can employ this pattern in order to handle each request in a single thread. Thus, the specific benefit of this pattern is its simplicity, as it does not require parallelism for single users and also no additional components. However, its simplicity is also its drawback, as it is not suited for complex grading tasks that may be long-running or consume many resources. In these cases, students may have to wait a long time for a system response or may even overload the server with requests.

Asynchronous Push

An alternative is the *asynchronous push* pattern, which also triggers the grading process directly, but without blocking user interaction by waiting. Instead, the actual grading task is started in a new thread that runs in parallel to processing further user interactions (see fig. 7.2). An example can be found in [315] with components offering interfaces to trigger grading and sending back grades, respectively. As for the previous pattern, either a queue or an evaluator component used in this pattern needs to be a passive service that waits for synchronous push requests.

The pattern has two variants, depending on how the system proceeds after the grading task has finished. One option is to push the result to the user interface component, which

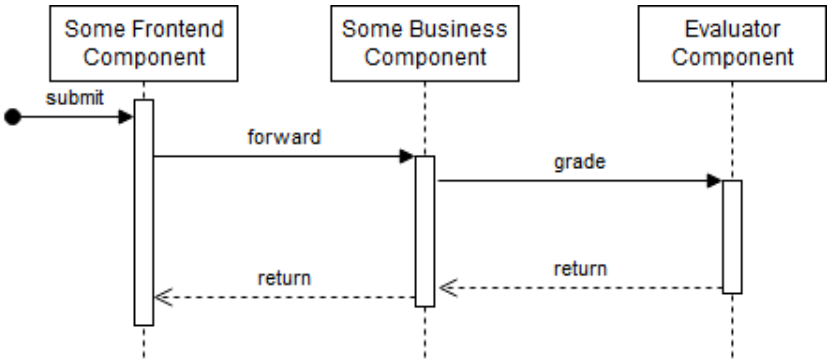


Figure 7.1.: Sequence diagram for the *synchronous push* pattern.

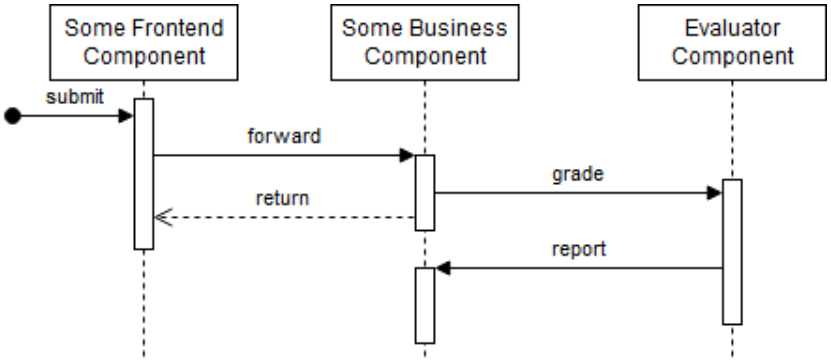


Figure 7.2.: Sequence diagram for the *asynchronous push* pattern.

may not be possible e. g. in a web-based system that does not support push notifications. The other option is to just store the result and deliver it upon a later request issued by the user interface component. Independent of the choice of a variant, the benefit of this pattern is that response times to students can be kept low. However, if the next meaningful page displayed to the student depends on the grading results, students will just receive a message asking them to wait. In addition, a risk of system overload still exists, as many students can trigger grading processes at the same time.

Asynchronous Pull

A third alternative is the *asynchronous pull* pattern, in which user input is stored in a queue and pulled from there by the evaluator component. This pattern requires to introduce an additional component to handle the queue of waiting student responses (see fig. 7.3). It also requires the evaluator component to be an active agent that can perform pull requests to the queue. Examples of this pattern can be found in [11, 269, 228], but also in many other places. Slight variants exist where one option is to put the whole submission to the queue and another option is to generate grading tasks and put those to the queue. Independent of the variant, the same two possibility as above exist on

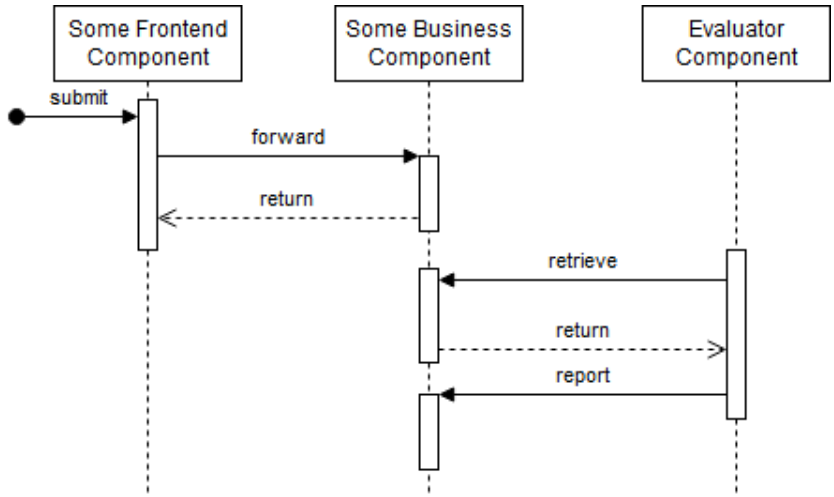


Figure 7.3.: Sequence diagram for the *asynchronous pull* pattern.

how to proceed after a grading task is finished. Notably, if grading is split into tasks, a push notification to the user interface can be triggered after completion of each task or after completing all tasks. Benefits and limitations are also very similar to the previous pattern: Response times are kept low, but students may just receive a waiting message as initial response. However, there is an additional benefit in that many responses at the same time are less likely to overload server resources. Instead, they will just fill the queue and increase wait times until they are processed. Notably, this pattern requires at least one active component within the system that frequently checks the queue for waiting grading tasks. The other patterns discussed before do not need such component and can thus also be used in systems that are composed of passive services and thus solely react on user input.

Asynchronous Push and Pull

A fourth alternative is the *asynchronous push and pull* pattern that combines the previous push and pull approaches to realize fully asynchronous communication. It introduces the queue component as a standalone element that is neither integrated into the business component nor the evaluation component: The business component pushes jobs asynchronously to the queue, and evaluator components retrieve jobs from it asynchronously as well (see fig. 7.4). Results can be delivered to the business component directly or stored in another queue and retrieved from it in a similar way. While that pattern has no impact on the performance in comparison to the previous pattern, it allows for flexible reuse of evaluator components in different contexts. Several business components (i. e. several instances of the same e-assessment system) can use a single queue. Evaluator components can thus serve different sources of submission and push their results back into the queue. This is particularly beneficial if there are specialised evaluator components which are only used rarely in peek times. Instead of providing

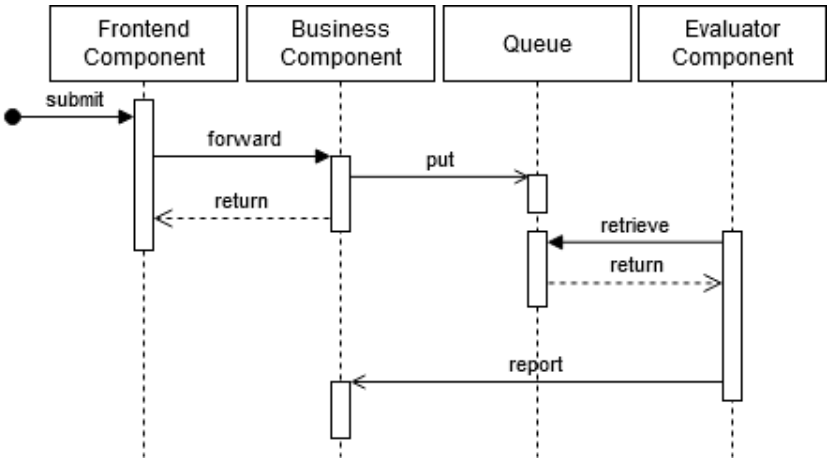


Figure 7.4.: Sequence diagram for the *asynchronous push and pull* pattern using direct delivery of results.

several evaluator components that are idle most of the time, a single one may be sufficient to handle jobs from several e-assessment systems.

7.2.3. Internal Evaluator Component Behaviour

The same structure of an evaluator component can be used in multiple ways with respect to parallel calls or parallelism within evaluation processes. The following patterns show solutions and their consequences that help to tackle more general challenges when used in combination with the other patterns in this catalogue. Some of the patterns are only relevant if the evaluator component uses sub-components that will be discussed in section 7.3.3 below.

Figure 7.5 provides a schematic representation of the four possible combinations of the patterns from this section. Boxes with vertical overlap represent parallel execution. As can be seen, variant (a) employs no parallelism and thus takes the most time, while variant (d) makes heavy use of parallelism and consumes the lowest amount of time.

Single-Threaded Evaluator

In the *single-threaded evaluator* pattern, the evaluator component handles only one evaluation job at once. In the simplest case, it is invoked in a loop over the list of waiting submission (as e. g. in [163]) that makes sure that there is only one call at a time. If the evaluator component can be invoked from different places, it may reject further requests or store them in a queue for later processing, depending on the communication pattern used (see section 7.2.2 above). This pattern provides a sufficient solution if an evaluator component handles only one job at a time and if the evaluation process is either fast in general or there is no need to make a complex process as fast as possible. No special attention needs to be paid to the use of shared resources when implementing the evaluator

Figure 7.5.: Schematic representation for parallel or sequential processing of jobs over time with the four possible combinations of patterns for evaluator behaviour. Variant (a) employs a *single-threaded evaluator* with *sequential evaluation*. Variant (b) uses a *single-threaded evaluator* with *parallel evaluation*. Variant (c) combines a *multi-threaded evaluator* with *sequential evaluation*. Variant (d) uses a *multi-threaded evaluator* and *parallel evaluation*.

or any of its sub-components following this pattern. It thus helps if strict isolation is required for security reasons.

In turn, the pattern limits the solution-space for speeding up the process, as it requires to deploy several evaluator instances to achieve parallel evaluation of multiple submissions. An example of automatic scaling by spawning additional instances of overloaded evaluator components is mentioned in [315]. The pattern also requires to implement some queue for waiting submissions if it cannot be guaranteed that there is always an idle evaluator instance available for each incoming evaluation job.

Multi-Threaded Evaluator

An alternative to the parallel deployment of evaluator component instances is the use of the *multi-threaded evaluator* pattern. In this pattern the evaluator can handle multiple evaluation jobs at once (as e. g. in [93]). It may either spawn a new thread for each incoming evaluation job or use a thread pool. Similar to the previous pattern, it may reject further requests or store them in a queue for later processing. The pattern requires

to consider the use of shared resources during the implementation and reduces the degree of isolation. In turn, it allows to speed up the evaluation process by handling several submissions at once. The actual effect of that measure depends on the underlying computing infrastructure and their capability to do efficient parallelisation. It may still be necessary to implement some queue for waiting submissions if it cannot be guaranteed that there is always the possibility to spawn another thread for a newly arriving evaluation job.

Sequential Evaluation

Independent of the choice for one of the previous patterns, an evaluator may have sub-components, that may or may not run in parallel even if the evaluator only processes one grading job at a time. In the *sequential evaluation* pattern, the evaluation process is organized in such a way that only one sub-component will be active at the same time. For example, [212] describes a system in which a sequence of sub-components is applied to a submission during the evaluation process. Notably, the actual decision which sub-component to invoke next can depend on the results from the previous sub-component. Thus, this pattern can particularly be considered if there are dynamic dependencies between several activities in the evaluation process. Each sub-component will know the full results of all preceding evaluation steps and can thus react accordingly. Moreover, the evaluation process can be adapted dynamically, potentially skipping steps if preceding steps meet specific conditions. On the downside, sequential evaluation can result in large wait-times for students if the evaluation process contains many time-consuming steps. Even running several processes in parallel will only increase the overall throughput, but not lower the time needed for processing one evaluation job.

Parallel Evaluation

Individual wait-times can be reduced with the *parallel evaluation* pattern, if there are several sub-components within the evaluator component. In this pattern, the evaluator invokes all or at least some required sub-components in parallel, waits for all of them to be finished and returns the collected results. This is possible, if there are little or no dependencies between the activities in the evaluation process. The main benefit of this pattern is to speed up the evaluation process for a single submission and thus lowering the wait-time for the student. On the downside, not all evaluation processes may be suitable for being split up into parallel tasks due to potential dependencies between sub-components. In addition, using parallel evaluation in an evaluator that also uses multi-threaded evaluation may result in a total amount of too much parallel threads so that actually no additional performance benefit is gained from multi-threading.

7.3. Structural Patterns

For the purpose of this chapter, structural patterns are patterns that define possible connections between components or the number of occurrences of components of a

particular type within an architecture. Furthermore we assume design clarity to be the main driver for decisions on structural patterns, where no decision should limit required system behaviour.

7.3.1. System Extensions

The most prominent way of extending an existing e-assessment system with new features is adding new types of assessment items. This usually requires to extend the existing user interface to allow for new interactions. In many cases, this also requires to extend the evaluation capabilities of the system in order to handle the new types of responses. In a component-oriented system, these extensions are not only possible by extending an existing component, but also by adding a new one and connecting it properly to the existing components. There are several ways of how to perform this integration.

One way of using components that offer specific features from other components is to use plug-in mechanisms. In a general plug-in mechanism, a host component `H` defines a plug-in API and interface. Another component `P` can implement the interface and make calls to the API. As a consequence, `P` can be used as a plug-in to `H`. It is left to `H` when to make calls to `P` and when to hand over control to `P`. Once control has been handed over to `P`, it may make calls to the plug-in API and of course may hand back control later. For example, `H` is a general learning management system which would like to offer specific exercise types suitable for assessments. In this example, `P` can be a component offering at least an user interface for one or more of the desired exercise types. `H` can then include `P` as a plug-in and call it whenever an user interface including the specific exercise types is requested. However, it heavily depends on the design of the plug-in API how tight the integration between `H` and `P` is.

Systems that do not allow for extensions will not use any of the patterns that are discussed in the following sections. If systems allow for extensions, they are not restricted in the number of ways they can offer and thus a system may use more than one of the following patterns. However, understanding a system's architecture becomes harder the more different patterns for system extensions are used.

Encapsulated Plug-In

One pattern is that of an *encapsulated plug-in*, which implements the full feature set of the new component. It is written in the same language as the existing system and uses the data storage and other components provided by the existing system. Usually there is one core component in the existing system offering an appropriate API to be used by plug-ins (see figure 7.6). This is the standard way of implementing plug-ins in learning management systems like Moodle (cf. [46]) or Ilias. The benefit of this pattern is that a well-written plug-in API can ease plug-in development and assure a close integration. At the same time, the API may also limit the plug-ins in what they do, which can protect the whole system and its users from malicious components. On the other hand, this may also be a downside if some sophisticated features cannot be implemented that way, e. g. if they require specific resources. In addition, a badly-written plug-in API may allow

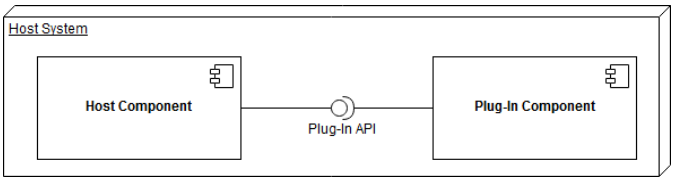


Figure 7.6.: Component diagram for the *encapsulated plug-in* pattern.

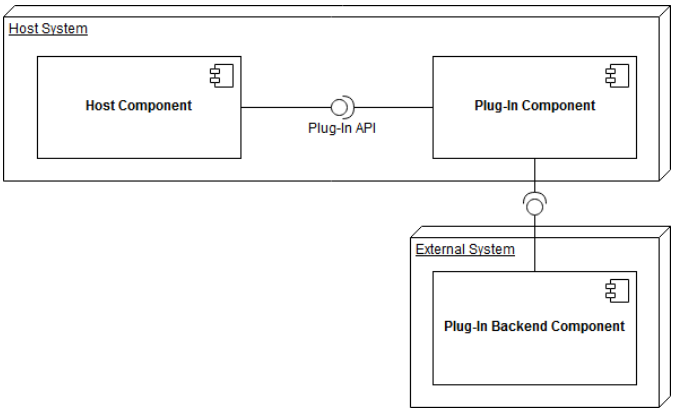


Figure 7.7.: Component diagram for the *unrestricted plug-in* pattern.

plug-ins to pollute the data storage of the system (e. g. by not cleaning up temporary data) and thus may cause problems in system maintenance or performance.

Unrestricted Plug-In

An alternative pattern is that of an *unrestricted plug-in*, which only implements a subset of the desired features directly. Besides connecting to the plug-in API of the existing system, it also connects to an own backend component or other external resources that implement the missing part of the feature set and that may also have its own data storage mechanism (see figure 7.7). This pattern can be found in learning management systems as with the Moodle External API [46]. It can also be found in systems in which evaluator components are connected via an API but are allowed to access external systems (as e. g. in [269]). This pattern overcomes the drawbacks mentioned above, as the backend component may be implemented in a different programming language, have access to additional resources, and store data in a separate place. This pattern can thus be used to integrate expert components (such as computer algebra systems) into e-learning and e-assessment system. A drawback of this solution is that potentially critical data may leave the system and remain in foreign places, even if it is deleted from the original system. There is also a risk that the backend component may become a bottleneck with respect to system performance, as increasing system resources for the e-assessment system does not necessarily also increase system resources for the external backend component.

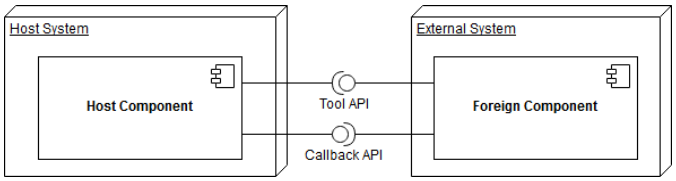


Figure 7.8.: Component diagram for the *external tool* pattern.

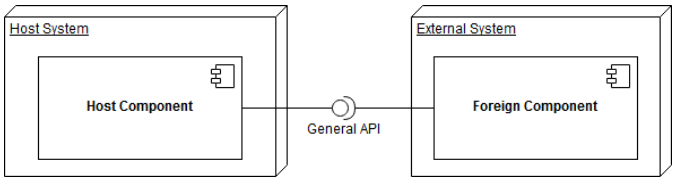


Figure 7.9.: Component diagram for a plug-in-free extension.

External Tool

The third alternative is that of an *external tool*. In this pattern, the existing system redirects the user to an external tool via some standard API and receives a callback when the user has finished their duties there (see figure 7.8). This mechanism can be realized in learning management systems via the IMS-LTI standard [130]. This pattern actually avoids extending an existing system, but adds functionality by coupling it with another system. Consequently, a potential benefit of this pattern is a minimal amount of integration work if both sides are already ready for integration via standardized protocols. The pattern also enforces a quite strict separation of data between the two sides. This can be beneficial and a limiting factor at the same time: On the one hand, it improves privacy and simplifies data management. On the other hand it requires to establish a trusted connection between the tools so that one tool accepts the user sign-ins from the other tool. In addition to these security and privacy issues, the drawback of a performance bottleneck mentioned above also exists in this pattern. Moreover, it is less likely to achieve a good visual integration of user interfaces via this pattern, as different tools may follow different user interface design principles.

Plug-in-free Extensions

Notably, there may also be situations in which a system can be extended without using any of these patterns. For example, a system may use the *asynchronous pull* pattern from section 7.2.2 and require all evaluator components reading from the queue to be *active agents* according to section 7.2.1. In this case, an additional agent can simply be added without the need for any kind of dedicated plug-in API offered by some component (see figure 7.9). Instead, having a queue from which other components can read forms an implicit plug-in API in this case. This is for example the way the xQueue from the MOOC platform edX works [79].

7.3.2. Data Storage

Potentially leaking critical data to external places or pollution of data storage by careless plug-ins (as discussed in the previous section) are not the only issues with respect to data storage that can occur in e-assessment systems. There is a general question on how to store data in such a system for several reasons. First, nature and structure of data may be very different. An e-assessment system may store user profiles, assessment items and responses, domain-specific knowledge used for grading and hint generation, and management or administrative data. Second, most data may be irrelevant for most features and components. An authoring tool will most likely only access assessment items, while domain-specific knowledge is only relevant to the respective evaluator component, pedagogical module, or problem generator. Third, there may be time-based restrictions on when to access which data. Changing assessment item may be prohibited while an exam is running, while teachers may be allowed to inspect student specific data only for a limited period due to privacy reasons. Consequently, there are different ways of how to handle data storage and system designers must decide which one to use.

Central Data Storage

The simplest way is to employ the pattern of a *central data storage* that accumulates data for all components. This is particularly useful when using several components that are supposed to work on the same data. Moreover, data storage can be centralized in cases in which a common service for data storage exists, as e. g. shown in [269, 315, 108]. A particular drawback of this pattern is the need for a generalized storage technology that is sufficient for all different kinds of data. This may lead to non-optimal solutions and potentially in a decrease of storage performance. On the other hand, a centralized data storage is easy to manage and can be replicated to multiple server nodes by standard technologies. Moreover, it allows to apply privacy concepts like differential privacy [76] that would be more complex to apply when multiple data sources are used.

Distributed Data Storage

An alternative pattern is that of a *distributed data storage*, which is used when components typically process specific data that is of no meaning to other components. Examples include separate storage for assignments, grades and general student data [212], the separation of learning repository and user model components [61] or the separation between a relational database for general data and a repository for submitted program code [87]. Another example similar to the latter is shown in [156], where different storage backends are sketched. The main benefit of this pattern is an increased performance, as each storage node can be optimized with respect to storage format and system resources. As a downside, this makes system management more complex. With respect to privacy, this pattern can enforce certain properties on system level (instead of application level), as data that must not be combined can be stored in different places.

Duplicate Data Storage

A third pattern is that of a *duplicate storage*, where data is prepared and stored in one place but copied to another place on demand. This is used for example when item pools are stored in one place for authoring and copied to another place when running an actual assessment as shown in e. g. [115]. Benefits and drawbacks of this pattern are similar to the ones for a distributed data storage. In addition, data synchronisation can become an issue when using duplicate storage. On the other hand, the pattern also allows for even more fine-grained control over which data is accessible by which component and when, so that even more privacy properties can be assured on system level. The pattern can be used in combination with a *distributed data storage*.

7.3.3. Evaluator Granularity

The most common educational component within an e-assessment system is an evaluator module that is responsible for grading responses and producing feedback. Section 7.2.2 already discussed different ways to invoke this component synchronously or asynchronously. This section deals with different patterns for structuring the evaluator component internally. Each evaluator component within a system will follow one of the patterns, but if there are many evaluator components, each of them may use a different one.

Monolithic Evaluator

A simple pattern is to design the evaluator component as a single block and thus as a *monolithic evaluator*. It receives a submission as input and returns grades and feedback as output. This is typically sufficient for short running synchronous grading tasks (e. g. [48, 108]). The complete behaviour of the component can be modelled as a single process in this case. To speed up grading in scenarios with high load, several of these processes can run in parallel, which will increase the overall throughput, but not lower the time needed for grading one submission. If entirely different grading procedures are necessary, two or more completely independent evaluator components can be employed following this pattern, as it is for example sometimes used for grading solutions in different programming languages [11, 205, 97].

However, if several different grading procedures are required that share some common elements, this pattern is not suitable, as it will either require to merge the grading procedures into one complex process or to duplicate code to use it in different independent components.

Evaluator with Sub-Components

A solution to the aforementioned problem is the use of sub-components within the evaluator component and thus forming an *evaluator with sub-components*. An example for this pattern can be found in [212], where the assignment database contains the sequence of modules to be applied to the submission upon reception by the grading server. Another example of this pattern with a different reasoning can be found in

[182]. In this case, the evaluator contains one sub-component that runs as a sandbox system for security reasons. A special case of this pattern occurs when domain-specific expert systems are used as sub-components for the evaluator (as e. g. in [106]). In this case, this pattern does not only determine the internal structure of the evaluator, but potentially also influences the connection between the e-assessment system and an external component. However, expert systems may also be available as libraries to be included into the evaluator component.

The pattern allows to build flexible evaluation processes in which sub-components can be used in different orders and combinations. It may also be sufficient to fulfill security requirements, but it offers less strict isolation than the *monolithic evaluator* pattern. The pattern may help to speed up the evaluation process, if sub-components can be invoked in parallel. However, this is subject to the use of respective behavioural patterns (see section 7.2.3 below). Notably, an evaluator with sub-components can be deployed in several instances without including the same sub-components in all instance. This allows to deploy dedicated instances e. g. for fast or memory-consuming evaluation steps, which also may help to speed up the evaluation process.

Evaluator as Re-director

A third alternative is the *evaluator as re-director* pattern, which is a mixture of the previous solutions. In this pattern the evaluator component has many sub-components but only uses exactly one of those for each evaluation job. An example of this pattern can be found in [269]. The pattern allows for more flexibility than a *monolithic evaluator*, since universal steps that are part of all evaluation processes can be implemented once in the evaluator component instead of multiple times in each sub-component. This can be particular beneficial in conjunction with some of the communication patterns discussed in section 7.2.2 above. In turn, isolation is less strict and the pattern does not help to distribute the evaluation processes over different nodes.

Similar to an *evaluator with sub-components*, not all instances need to include the same sub-components. Using such deployments is particularly advisable if some sub-components bear a risk of causing system crashes under some circumstances. Such crashes may stop the evaluator component from working at all and thus also stop all of its sub-components from delivering results. If there are several evaluator components and not all of them have the same risk of crashes, a higher overall availability can be achieved.

7.3.4. Input Processing

One of the most obvious functions of an e-assessment system is to process user input in order to provide feedback. Typically, evaluator components and pedagogical modules are the responsible components for this purpose and it may well happen that more than one such component is present in a system. In conjunction with the fact that an e-assessment system may or may not be client-focused or server-focused, there are several patterns that define the connections to those components.

Server-side Processing

For general distributed systems, *server-side processing* is a usual pattern, as input processing is typically associated with business logic functionality and business logic components in turn are typically located on the server-side of a distributed system. In the case of e-assessment systems, this means that at least one evaluator component or pedagogical module is only available on the server side of the system and is designed as a passive service. User input is forwarded from the client to the server and that invokes the appropriate component. Results from this component are then transferred back to the client. While this pattern is used in many systems as a plain design choice, it is the foundation of service-based and cloud-based e-assessment systems [11, 115, 315].

Several benefits arise from using this pattern. First and most important, this pattern is the only one suitable for strict, legally relevant e-assessment situations, as any grading operations on the client side must be considered inherently insecure. Second, this pattern is very helpful when grading and feedback generation requires to run specific software that may not be available on the client side at all or cannot be integrated in the client components of an e-assessment system. This is particularly relevant if grading requires large amounts of system resources but the system must be accessible from mobile devices with limited resources or if grading requires to access large repositories or databases that are not available online. Third, server-side processing makes grading and feedback data immediately accessible to teachers, potentially even in a larger extent than for students.

On the downside, scalability may become an issue if a large amount of users must be served with limited server resources. In this case, server-side processing can create a serious bottleneck. Moreover, server-side processing limits the options to use the client components of the system offline. While it may be possible to work offline on the actual assignment, every request for feedback requires an online connection to the server. Depending on the scenario, privacy may also be an issue for server-side grading, because sending submissions and grades via a network may expose personal data, even if no grades are stored in a server-side database.

Client-side Processing

Inversely to the previous pattern, *client-side processing* in the context of e-assessment systems means that grading or feedback generation for user input happens by invoking an evaluator component or pedagogical module directly on the client-side without forwarding user input to the server. Results from the grading process may be forwarded to the server as well for later analysis by a teacher, but this may be an optional function. There even may be no server at all, as for example in a peer-to-peer e-learning framework with some feedback agents on every peer application [201].

Benefits and drawbacks are also inversely to the previous pattern. As a first benefit, server resources cannot cause bottlenecks, when client-side processing is used. This applies both to computing resources and to delays from a slow or unstable network connection. Consequently, this pattern is particularly useful when fast and direct response to a user

input is required as for example in a listening agent [144]. Second, less issues with privacy have to be expected, as only limited data or even no data at all leaves the client.

However, this may also be a serious drawback of this pattern, as detailed analysis of assessment results by a teacher looking at the server data may not be possible if some data is only available on the clients. Moreover, any data produced on the client side may be subject to manipulations before it is send to the server, and so any reliable data analysis is not possible when this pattern is used. Finally, client-side processing may result in too high or complex requirements for the client system to be efficient.

Mixed Processing

Some of the drawbacks of client-side or server-side processing can be avoided without introducing all the drawbacks of the opposite pattern by using *mixed processing*. In the mixed processing pattern, one grading or feedback generation process is split in a way that it is handled partially on the client side and partially on the server side. Notably, this is not the same as having both patterns applied to the same system for different processes.

Mixed processing can be used to perform simple but mandatory process steps on the client, while the server is only invoked for some optional, but complex or resource demanding operations. This reduced the risk of overloading the server with simple task and at the same time reduces the need for powerful computing resources on the client side. Similarly, this can be used to perform some calculations on the server to avoid manipulations, but at the same time keep some data on the client to avoid privacy issues. As a general drawback, mixed processing increases system complexity. Hence it may be worth the effort to consider splitting a larger grading or feedback generation process into smaller ones where each of those can be realized either using client-side or server-side processing. This is a similar decision as about using one multi-threaded evaluator or invoking several single-threaded evaluators in parallel.

7.4. Functional Patterns

For the purpose of this chapter, functional patterns are patterns that define the actual behaviour of specific components and which components need to be present and interconnected to realise a particular feature. As for the behavioural patterns discussed earlier, we again assume that a particular required system behaviour is the main driver for decisions on functional patterns for a given scenario.

7.4.1. Adaptive Behaviour

Adaptive behaviour is particularly common in intelligent tutoring systems, but also used in e-assessment systems, e. g. to make assessments more efficient or more motivating. Adaptation in e-assessment systems can happen in several different places: (1) The system can present different items within an assessment by assembling the assessment dynamically using an assessment generator component; (2) the system can present the

same items with different content using an item generation component; (3) the system can present the same item but offer different interactions if the item itself is adaptive. This section focuses on the mechanisms for adaptivity that decide when to adapt and which data to use for the decision on how to adapt. The actual process of adapting item content is considered in the next section below.

Path-based Adaptivity

The basic concept of the *path-based adaptivity* pattern is to use a state machine model where some events trigger transitions that change the current state. The system behaviour is fixed for each state and hence the transitions and their trigger events determine the adaptive capabilities of the system. While this can directly be used to branch system behaviour based on user interactions, this also implies some history mechanism. Let state S have several outgoing transitions to different states T_1 to T_n, that all show the same behaviour. Then from a short-term user perspective, any interaction i with state S causes the same system reaction. However, T_1 may be followed by state U, which is not reachable by T_2 to T_n. So the fact that a users reaches U after T_1 is not determined by the interaction with T_1, but by the previous interaction i in state S.

An example on how to apply path-based adaptivity while using a rule-based reasoning system can be found in [244], where multiple-choice and single-choice items are modeled as finite state machines. In very simple cases, this only defines different behaviour for right and wrong answers, which is the expected default behaviour for e-assessment systems and can hardly be considered adaptive behaviour. However, this can also be used to react differently to the first try and a second try (as e. g. realized by the adaptive item format in the QTI standard [131]) or to proceed or grade differently depending on user input in multi-step items [248, 23]. The pattern cannot only be applied on item level zo present different steps within one item, but also on test level to present different items within one test. It can also be used on even higher levels to present different tests or to recommend different learning materials based on the outcome of a test. In that case, path-based adaptivity happens on the level of a system that has integrated e-assessment as one of its features as for example with the learning paths in Moodle.

If rules for triggering transitions are kept simple in the sense that they only evaluate data which is available anyway, no sophisticated additional components need to be introduced in the system to apply this pattern. Instead, item definitions, test definitions or alike need to be extended by appropriate state machine descriptions. A limitation of this pattern is the necessity to foresee possible different interactions to create proper triggering events for transitions. A system using path-based adaptivity will thus never show behaviour that has not been explicitly described.

Matching-based Adaptivity

Different to modeling explicit states, the *matching-based adaptivity* pattern models student properties and target properties as separate (mostly numeric) models and then adapts system behaviour based on appropriate proximity metrics using such models. For example,

student capabilities can be modeled as numeric values in one or more dimensions and item difficulties can be modeled the same way. An assessment generator can then select the most appropriate next item dynamically in an adaptive assessment [301, 114]. Student modeling and matching-based adaptivity is not limited to capabilities and competencies, but can also include other factors such as affective states [232, 261, 201].

Using this pattern requires not only to have an assessment generator or problem generator component to perform adaptivity, but also to have a student model component to store and manage the necessary data following one of the various possibilities of doing so [40, 71]. The main benefit of this pattern is that there is no need to foresee and encode all possible ways to adapt. Instead, the matching algorithm will find the best possible match even in unexpected situations. However, creating an useful student model and difficulty model usually requires a large amount of data. Hence, this pattern is not appropriate in situations where only a small data base is available.

Similar matching techniques as the ones used for adaptivity can also be used for feedback generation in some domains where solution spaces exist that can be represented as models. A more detailed discussion of this feature is deferred to part III of this document, as it influences the internal design of domain-specific evaluator components but has no direct implications for the general system architecture.

7.4.2. Item Generation

Item generation (also named "problem generation" in some contexts) is the process of producing individual assessment items dynamically instead of selecting static items from an item bank. If an e-assessment system should be able to do so, usually an item generator component is included in the system. The following sections discuss several patterns on how the item generation process can be designed from a purely technical perspective. Part III of this document digs deeper into the details of the domain-specific creation of item contents from other perspectives such as usage of specific tools or data formats. All patterns discussed in this section can be used both for closed or open-ended questions and hence do not make any implications for the grading of a dynamically generated item. For illustration, single-choice-items will be used to explain the different patterns throughout this section.

Templates with Selections

In the *templates with selections* pattern, items consist of content templates that contain placeholders and lists of possible values for each of these placeholders. In a slight variant, single-choice-items and multiple-choice-items can draw their answer options randomly from a pool. For some types of values such as numbers it is possible to define a range instead of listing all possible values [100]. Besides using this technique to replace words or numbers within a text as done by many systems, it can also be used more creatively to fill tables or change size scales in technical drawings [139]. The list of possible values needs not to be encoded directly in the item definition, but can also be retrieved from an external data source [88]. Generating an item instance means to select one of the

possible values for each of the placeholders. This selection can happen purely random or based on some rules that express dependencies between different placeholders.

A sample single-choice-item using templates with selections may name a country and ask for its capital by offering several city names as answer options. The country name and the city names will be placeholders in this case, each associated with appropriate lists of names. Selecting a country name can happen purely random, while selecting city names includes dependencies to assure that the correct city is shown among the answer options and no city appears twice. In some cases using fill-in items it may not even be necessary to define dependencies for correct answers, as an evaluator component might be able to deduce them directly from the item parameters [41].

The main benefit of this pattern is its universal applicability. As the selection process is random or follows some simple rules, it does not need to know anything about the item content and hence can be used in any domain. There is also no need to include additional components in the item generation process. Moreover, item templates are easy to understand as long as the lists of options and dependencies are not too large. This in turn is also a limitation of this pattern, as not all scenarios can be expressed in a manageable way using lists of values and dependencies. For example, it is at least tedious to express the random generation of a polynomial via random selection of coefficients and the generation of its first derivation via dependent placeholders.

Templates with Computation

As an alternative to purely random selection and the evaluation of dependencies, the *templates with computation* pattern can be used to introduce arbitrary computations when determining placeholder values. These computations may include domain-independent operations such as string concatenation or basic mathematical operations [131, 23], but also complex operations performed by external domain-specific expert systems such as encrypting a text [213], computing a graph layout [39] or computing the derivation of a polynomial.

In a sample single-choice-item this may result in providing a randomly generated polynomial and offering one correct and some wrong derivations as answer options. Assuming that an appropriate expert system or domain knowledge repository is available, virtually any problem generation technique can be used with this pattern, including those that were originally not created in the context of e-assessment (such as [251] for DFA construction problems, [113] for statistics, or [4] for the general generation of multiple-choice items from semantic relations).

This pattern avoids the limitations for templates with selections and thus makes item generation more powerful. It also requires more domain knowledge from an item author, but this is not a limitation. Instead, it can actually be a benefit as an item author can express content generation directly using computation concepts from the domain (possibly supported by a domain-specific notation such as in [295]) instead of transforming them into a set of selection dependencies. The potential need to use external expert systems such as computer-algebra-systems may cause limitations in terms of performance

if computing resources for such systems are limited and item generation requires many calls to those systems.

Content Construction

Instead of using item templates defined by an item author, there is also the possibility to use the *content construction* pattern. This pattern uses more general item templates and fills those based on a domain knowledge models [6, 132] or some other appropriate input [161, 153]. As a result, it is not just some portion of the item content that looks differently for several item instances, but the item may only keep its form. For example, the content construction process for a single-choice-item may use a domain knowledge model on ancient Roman emperors and ask "How was the successor of emperor X?" in one case and "At what age did emperor Y die?" in another case.

This pattern reduces the necessary size of an item definition to a minimum. In particular, it enforces a separation between the definition of items and the encoding of actual facts. Consequently, it requires to have a domain model component or a parser for other appropriate domain artifacts available within the e-assessment system or as an external service. As for the templates with computation pattern, this may cause limitations in terms of system performance.

7.5. Pattern Summary

A total amount of 28 patterns for eight different aspects in three categories have been discussed in this chapter. Tables 7.1 to 7.3 provide overviews on all patterns and name the participating components for each pattern according to the component names from section 6.1. While some of the different aspects are independent of each other, some others have dependencies. Choosing a particular pattern for one aspect may thus narrow the space for design decisions on another aspect. The tables thus also list possible dependencies or restrictions between different patterns.

The fact that the patterns *Passive Services* and *Active Agents* can be applied to any component and imply no dependencies or restrictions supports the fact that they were classified as patterns for general component behaviour. The same unlimited use can be observed for the patterns *Encapsulated Plug-Ins* and *Unrestricted Plug-Ins* which is also conclusive, as they represent very general mechanisms for extending systems. Finally, client-side processing can also be used with any component and without any restrictions which is also conclusive, as it basically means to develop an e-assessment system as a standalone application.

Besides some self-evident connections between specific components and groups of patterns (such as the involvement of the evaluator component in all patterns for grading a response and evaluator granularity) there are also some connections that allow to refer back to the discussion on some specific components. In particular, a data transfer component, which occurred rarely in the literature review, is mandatory for the *Duplicate Data Storage* pattern and hence it can be concluded that this pattern is also used rarely.

Aspect	Pattern Name	Participating Components	Dependencies and Restrictions
General Component Behaviour	Passive Services	any	none
	Active Agents	any	none
Component Communication for Grading a Response	Synchronous Push	Student Frontend or Queue, Evaluator Component	Queue or Evaluator Component as Passive Service
	Asynchronous Push	Student Frontend or Queue, Evaluator Component	Queue or Evaluator Component as Passive Service
	Asynchronous Pull	Student Frontend or Queue, Evaluator Component	Evaluator Component as Active Agent
	Asynchronous Push and Pull	Student Frontend, Queue, Evaluator Component	Student Frontend and Evaluator Component as Active Agent
Internal Evaluator Component Behaviour	Single-Threaded Evaluator	Evaluator Component	none
	Multi-Threaded Evaluator	Evaluator Component	none
	Sequential Evaluation	Evaluator Component	Evaluator with Sub-Components
	Parallel Evaluation	Evaluator Component	Evaluator with Sub-Components

Table 7.1.: Summary table for behavioural patterns for e-assessment systems

Aspect	Pattern Name	Participating Components	Dependencies and Restrictions
System Extensions	Encapsulated Plug-Ins	any	none
	Unrestricted Plug-Ins	any	May imply Distributed Data Storage
	External Tool	any	Enforces Distributed Data Storage
	Plug-in-free Extensions	any	Extensions as Active Agents
Data Storage	Central Data Storage	any storing component	none
	Distributed Data Storage	any storing component	none
	Duplicate Data Storage	any storing component, Data Transfer Component	none
Evaluator Granularity	Monolithic Evaluator	Evaluator Component	none
	Evaluator with Sub-Components	Evaluator Component, Expert System (optional)	none
	Evaluator as Redirector	Evaluator Component, Expert System (optional)	none
Input Processing	Server-side Processing	any	At least one Passive Service on the server-side
	Client-side Processing	any	none
	Mixed processing	any	At least one Passive Service on the server-side

Table 7.2.: Summary table for structural patterns for e-assessment systems

Aspect	Pattern Name	Participating Components	Dependencies and Restrictions
Adaptive Behaviour	Path-based Adaptivity	Student Frontend or Assessment Generator	none
	Matching-based Adaptivity	Student Frontend or Assessment Generator, Student Model	none
Item Generation	Templates with Selections	Item Generator	none
	Templates with Computation	Item Generator, Domain Knowledge Model or Expert System (optional)	none
	Content Construction	Item Generator, Domain Knowledge Model or Expert System	none

Table 7.3.: Summary table for functional patterns for e-assessment systems

Similarly, the existence of a student model component is directly coupled to the usage of the *Matching-based Adaptivity* pattern for adaptive behaviour.

Most dependencies and restrictions refer to the patterns *Passive Services* and *Active Agents* and thus implicitly encode a decision on which component is the leading part in a process. The only exception to that observation is the *External Tool* pattern for system extensions, which enforces to use the *Distributed Data Storage* pattern and thus explicitly defines that there is no leading system in terms of data storage.

8. Case Studies

The purpose of this chapter is to apply the pattern catalogue and conceptual framework developed in the previous chapters to some e-assessment system architectures that are documented in literature. The goal is to describe these architectures using the terms and concepts developed above. A particular focus within these case studies is put on integration aspects to be in line with the overall topic of this publication.

8.1. Case 1: JACK

JACK is an e-assessment system that has been started as a system for assessing programming exercises in Java in the year 2006. It has evolved to a multi-domain e-assessment system in recent years with some involvement of this publication's author in the architecture development and system implementation. The architecture of the system is primarily documented in [268] and [269].

The general architecture of JACK (see fig. 8.1) is a layered architecture with a design focus on the server-side. The client-side is realized via web-pages delivered by the server and a single plug-in for the IDE Eclipse that only eases downloading assignments and uploading submissions for students in specific exam situations. All relevant input processing happens on the server-side. The server-side is logically split up into a core server and a worker server, where multiple instances of the latter can connect to one instance of the former. The core server consists of passive services organized in two main layers, where one layer connects to the frontend (web and web service for the Eclipse plug-in) and the other one offers business logic capabilities and connects to the data storage. The data storage follows the *Centralized Data Storage* pattern and uses a single relational database.

With respect to grading, JACK employs the *Asynchronous Pull* pattern in combination with a *Evaluator as Re-director* for grading programming assignments. The worker server employs an active agent called "worker core" that has a set of different sub-components as so-called "checker plug-ins". The grading process for a single submission is split up into several jobs right after submission by the business logic component of the server core. The worker core thus is able to fetch one of these jobs from a queue and forward it to an appropriate checker plug-in. If necessary, one checker plug-in can call another one. Once the checker plug-in has finished, the worker core returns the result and has thus finished the grading process for the respective job. If a single submission produced several jobs, these can either be processed by one worker core one after another or by several worker cores in parallel in no guaranteed order.

In addition to that, JACK also uses the *Synchronous Push* pattern in combination with a *Evaluator with Sub-Components* for some short-running grading tasks. This

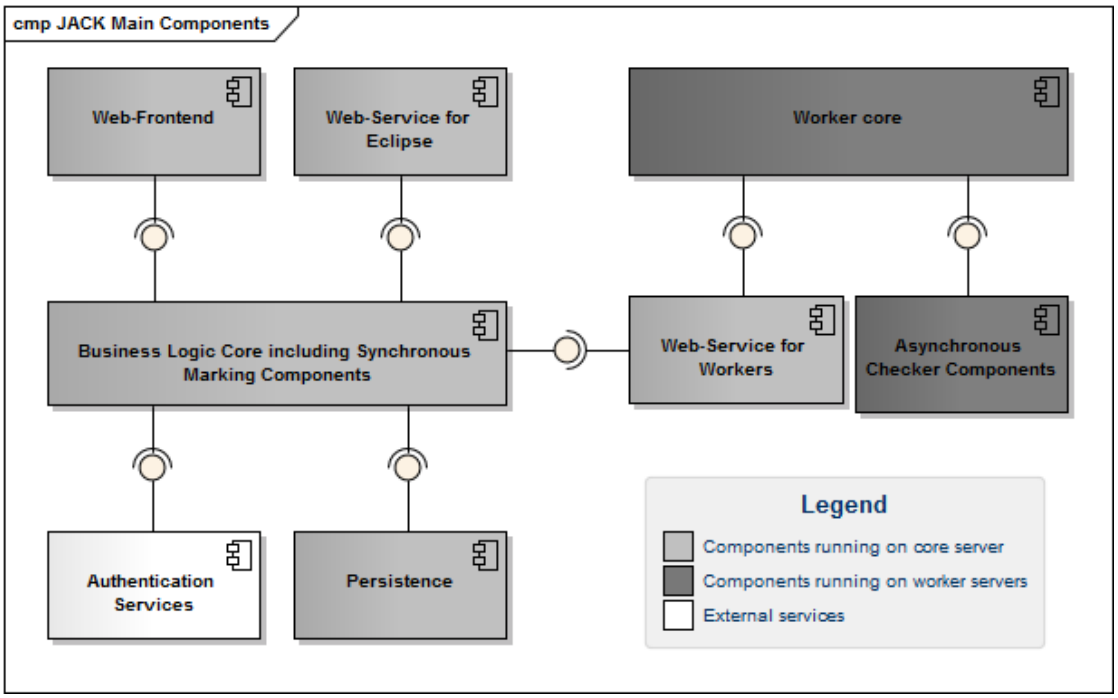


Figure 8.1.: Architectural overview of JACK showing all main components. One core server can serve multiple worker servers and one worker server can connect to multiple core servers.

combination is used for so-called "form-based" assignments, such as multiple choice or fill-in items in which only a moderate complexity for evaluations exist. In particular, the evaluator components for these exercise types may connect to expert systems such as computer-algebra-systems to perform evaluations. The evaluator component is a *Multi-Threaded Evaluator* that performs *Sequential Evaluation*. The same mechanism with calls to the same expert systems is also used for item generation for these types of assignments, where JACK uses the *Templates with Computation* pattern. JACK does not maintain an explicit domain knowledge model, but uses an item generator within the business logic that can connect to the same computer-algebra-systems that are used for evaluation. In addition, JACK uses the *Path-based Adaptivity* pattern on these types of assignments and thus allows to dynamically adapt the sequence of steps in a complex multiple-choice and fill-in item.

Several different patterns are relevant for JACK's capabilities in system extensions and system integration. The worker server uses the *Unrestricted Plug-In* pattern for the checker plug-ins that realize different checks on programming assignments. Each plug-in may realize a particular kind of check for a particular programming language and is allowed to invoke arbitrary sub-components or external resources to do so. In particular, each checker plug-in has to be written in Java and to implement an Java interface defined by the worker core, but may invoke other components that are compatible with the target

programming language of the submission under test. This is for example used when checking programming assignments in C++ [122].

As the worker core within the worker server part of the architecture is an active agent, it can also be used as a *Plug-in-free Extension* to systems offering an appropriate API. An actual example that has been used with JACK in this configuration is the xQueue from the MOOC-platform EdX [79]. Finally, the server core of JACK implements the tool provider part of the IMS LTI standard and can thus be used within other systems following the *External Tool* pattern.

In summary, JACK is a system that makes some efforts to provide a clearly defined architecture and to use and offer appropriate integration mechanisms. Grading for form-based exercises and programming exercises is implemented in an entirely different way and with different features so that the integration mechanisms seem to work well at least for practical purposes. A variety of patterns could be identified and used to describe the key concepts of the system, so that also the pattern catalogue and the conceptual framework seems to be sufficient for this case study.

8.2. Case 2: ActiveMath

ActiveMath is an intelligent tutoring system for mathematics that includes an extensive exercise subsystem that is documented in many details in several publications [105, 106, 107].

The general architecture of ActiveMath is a layered architecture with a focus on the server-side. A presentation engine on the first layer provides several views as browser-based user interfaces. An "exercise subsystem", a "tutorial component" and some data stores form the second layer of the general architecture. All components are passive services that are triggered by user interactions to perform grading solely on the server-side. Within that architecture, the *Distributed Data Storage* pattern is used to separate between a knowledge base for domain knowledge, a storage for learning materials called "books" and stores for strategies used within the exercise subsystem. Notably, a visual authoring tool for exercises in ActiveMath is available, but as there is no automated data transfer or synchronization between that tool and the core system there is no reason to assume the existence of a duplicate data storage.

Grading in ActiveMath happens according to the *Synchronous Push* pattern and employs a *Evaluator with Sub-Components*. The user interface forwards an user response to an item via the "exercise controller" and the "exercise manager" to the "diagnoser" that is the main evaluator component within ActiveMath. The diagnoser can use a "query broker" to issue calls to external expert systems such as computer-algebra-systems or domain reasoners to analyse the student input. Although it is not described explicitly in any of the publications, the evaluator component seems to be a *Multi-Threaded Evaluator* that uses *Sequential Evaluation*. This is also a reasonable assumption, because performance data of ActiveMath allows for 100 to 150 parallel users, which is similar to Jack that uses the same combination of patterns for a similar task. Anyway, the result of the evaluation process is returned to the exercise manager that in turn decides

on how to proceed in the exercise presentation. In particular, it may involve an "exercise generator" that consults a student model and thus allows both for *Path-based Adaptivity* and *Matching-based Adaptivity*.

The exercise generator in ActiveMath is a very important component that not only allows for adaptation of steps in complex exercises, but also is able to generate hints based on domain knowledge provided by domain reasoners. For that part of its features it uses the *Content Construction* pattern, although it is not able to generate entire items based on a domain model. For example, it can use a domain reasoner for linear, quadratic and higher order equations to automatically construct a sequence of correct steps to solve a particular problem defined by an exercise author. The same domain reasoner is then also used in grading a response to find out whether a student input is a correct step towards a complete solution. For the generation of item contents, ActiveMath also uses the *Templates with Selections* pattern with simple randomization.

ActiveMath offers no specific interfaces or patterns with respect to system extension. Any external systems such as computer-algebra-systems and domain reasoners are included by explicit connections via the query broker and no patterns are present that define a unified way of adding additional systems. ActiveMath also does not offer specific interfaces to be included into other environments.

In summary, ActiveMath is a system that has been developed with a clear focus on a sophisticated feature set and extensive use of domain-specific tools, but without considering integration as a key characteristic on the architecture level. That gets clearly visible by the fact that almost every functional pattern from the pattern catalogue can be found in the system, but none of the patterns for system extension. At the same time, the pattern catalogue and the conceptual framework that were created with the focus on integrated e-assessment in mind turned out to be sufficient to describe large parts of a more closed tutoring system as well in this case study.

8.3. Case 3: The "Ultimate" E-Assessment System

In 2009, Armenski and Gusev provided thoughts on the architectural style and a conceptual description of the overall architecture of an "ultimate" e-assessment engine [19]. The publication was driven by the ideas of service-oriented architectures and a perceived need for a better integration of e-assessment systems into general web-based technology-enhanced learning systems such as learning management systems.

The overall architecture is designed as a layered architecture consisting of a presentation layer to connect to frontend components, a common services layer that connects to external services e. g. for authorization, an e-assessment services layer representing the business logic and a composite services layer. The latter manages connections to external content provides and also offers services to systems that want to include e-assessment services from the "ultimate" e-assessment system. Consequently, the architecture uses the *Passive Services* pattern and assumes all grading activities to happen on the server-side. The common services layer allows to use the *Distributed Data Storage* pattern and the

architecture includes an additional "content repository", thus distributed data storage seems to be the intended strategy.

Among the common services sketched in the overall architecture there is also a "service registry". This allows to use the *Plug-In-free Extensions* pattern to add additional functionality via additional services that register themselves and are discovered by other services as needed. At the same time, the composite services layer may offer appropriate services to other systems so that the e-assessment system can be used by these systems following the *External Tool* pattern.

Since the publication only provides an overall architecture, no details are given about the design of the evaluation processes and the design of evaluator components. Hence no patterns for evaluator granularity and grading a response can be identified from the architecture description. The same is true for functional patterns, which is not surprising as a conceptual description of an architecture is not expected to include functional details.

In summary, the architecture for the "ultimate" e-assessment engine presents a good complement to the architecture of ActiveMath as discussed in the previous section. The "ultimate" e-assessment engine is explicitly designed with interoperability and integration aspects in mind and thus presents no functional details. Notably, the design is primarily based on the concept of independent services and all patterns that could be identified are direct derivations from that principle. The system design does not explore the needs of different possible functionality and thus also does not answer the question on whether the "ultimate" design is able to fulfill all possible needs. As the design thus does not include some new or rarely discussed ideas, the pattern catalogue and conceptual framework was again sufficient to describe most of the architecture in this case study.

8.4. Case Study Summary

In this chapter, three different system architectures have been described using the pattern catalogue and the conceptional framework that was developed in chapter 7. In all three cases, the existing set of patterns was sufficient to describe the core characteristics of the architectures. Hence, it can be concluded that the pattern catalogue indeed covers most of the important patterns that are used in the design of e-assessment systems. Table 8.1 provides a summary of the different patterns found in the three case studies.

Although three case studies can hardly represent the full state-of-the-art in the design of e-assessment systems, several observations can be made from this table. First, using server-side processing and passive services seems to be a dominant design choice that was made in all three cases. Second, two systems (JACK and ActiveMath) are already enough to encounter all three possible patterns for item generation. Third, integration and extensibility are polarizing aspects, as JACK uses three out of four possible patterns for that aspect, while ActiveMath uses none. That stresses the intention of this publication to look at the design of technology-enhanced learning systems primarily from the perspective of integration.

Aspect	JACK	ActiveMath	The "Ultimate" e-Assessment System
General Component Behaviour	Passive Services, Active Agents	Passive Services	Passive Services
Component Communication for Grading a Response	Synchronous Push, Asynchronous Pull	Synchronous Push	none
Internal Evaluator Component Behaviour	Single-Threaded Evaluator, Multi-Threaded Evaluator, Sequential Evaluation	Multi-Threaded Evaluator, Sequential Evaluation	none
System Extensions	Unrestricted Plug-Ins, External Tool, Plug-in-free Extensions	none	External Tool, Plug-in-free Extensions
Data Storage	Central Data Storage	Distributed Data Storage	Distributed Data Storage
Evaluator Granularity	Evaluator as Redirector, Evaluator with Sub-Components	Evaluator with Sub-Components	none
Input Processing	Server-side Processing	Server-side Processing	Server-side Processing
Adaptive Behaviour	Path-based Adaptivity	Path-based Adaptivity, Matching-based Adaptivity	none
Item Generation	Templates with Computation	Templates with Selection, Content Construction	none

Table 8.1.: Summary table for the three case studies from chapter 8. The table indicates which patterns are used by which system for the different aspects of system architecture.

9. Results

The goal of the previous chapters was to elicit specific features and typical building blocks of e-assessment systems and to provide several static and dynamic views on e-assessment systems that can be used as a conceptual framework in terms of architectural patterns. Two catalogues are the main results from this effort. Since these results have been validated by three case studies, this chapter can now draw some conclusions with respect to the essential qualities e-assessment systems must ensure by design and aspects that need further research both in integrated e-assessment and beyond. In addition, some connections to part I of this publication can be drawn.

9.1. Contributions to Integrated E-Assessment

Two catalogues have been developed in the previous chapters: One catalogue of components for e-learning and e-assessment systems and one catalogue of patterns on how to integrate these components with each other. Both catalogues can be used to discuss the integration of e-assessment capabilities into other systems based on a unified terminology with respect to the purpose of the components. They can also be used to describe the technical integration within an architecture based on a unified terminology with respect to software design. Finally, they can be used to identify suitable places for integration interfaces within an existing system architecture that has originally not been designed with integration capabilities in mind. All these actions can be considered helpful to improve the design of e-assessment systems and to promote research and development in the area of e-assessment integration.

Three small case studies were sufficient to demonstrate how necessary more research in this area is indeed. On the one hand, one system analyzed in these case studies showed a rich amount of sophisticated functional features, but no characteristics for being integrated into other systems or offering interfaces for easy system extension by integrating other systems. On the other hand, conceptual ideas for e-assessment architectures exist and have also been discussed in the case study that focus on integration without digging into the details of e-assessment functionality. However, these architectural concepts may be too focused on a particular style and thus miss the capability to integrate sophisticated features. This is another lesson from the case studies, in which the analysis of two systems was sufficient to identify a total amount of five different patterns for functional aspects such as adaptive behaviour and item generation. Integrated e-assessment architectures will thus most likely always need to be open for integrating emerging approaches that add valuable features to existing e-assessment systems.

In turn, the results from the previous chapters can help to classify upcoming research results and provide silos in terms of well-defined components in which specific functionality

can be realized independent of the surrounding system. This can help to shape the design of new features and make them more universally applicable right from the beginning. New features should not be defined and understood as additions or amendments to a single existing system, but designed as improvements or evolutions of existing components or as completely new components not yet covered by the component catalogue. This way, it is much easier to estimate the benefits of these features and identify all potential context in which they can be used.

This also makes a strong connection to the results from part I of this publication. Components like "assessment generator" and "evaluator component" from table 6.2 directly link to activities such as "author tests" and "create feedback" from table 2.3 and even more closely to the states "generated" and "evaluated" for the alpha "Test" (see table 3.2). Consequently, the results from the previous chapters help to relate system features and behaviour to states and activities in the assessment process. This helps to identify system components or features that must be included into a system that is supposed to support a particular type of assessment process. In turn, this also helps to identify missing interfaces or patterns for component integration, if states are grouped in a process model that refer to components with limited integration capabilities. Similarly, we found commonly used user interfaces for participants and organizers, but got no commonly used interface component for authorities. This is in line with the fact that the latter are optional in the process, but also indicates that there might be a lack of integration of software components into the actual processes.

All these potential use cases for the two catalogues point to the same general findings: To enable integrated e-assessment, e-assessment systems must provide well-defined interfaces both on the level of system integration with external systems and on the level of feature integration within a system. The component- oriented view on system design made it possible to define patterns on how to integrate specific features like adaptivity, item generation or hint generation into a larger system context. The same design principles also explain how to integrate a particular system offering these features into the larger context of a learning management system or similar. If the goal is to integrate specific and sophisticated e-assessment features into general purpose learning systems, an approach based on well-defined components and reusable patterns seems to be a promising way to go.

9.2. Contributions beyond the Scope of this Publication

The component and pattern catalogues from the previous chapters are not strictly limited to the aspect of e-assessment integration. Instead, they can be used as a starting point for a general decomposition of an existing system into components. This can be helpful for describing an existing architecture that is already component based. It can also serve as a starting point for the re-design of an existing system that is not yet designed in terms of well-defined components.

The component and pattern catalogues can also be used as a universal basis to describe and compare e-learning and e-assessment systems in various types of research such

as literature studies or comparative analyses. The same is true for the connections between system architecture and assessment processes from part I of this publication. These connections also cannot only be explored with respect to e-assessment integration, but also more generally with respect to tool support and automation in e-assessment processes. For example, a mapping from activities to components can be created to find out whether some common activities are not yet supported by typical components of technology-enhanced learning systems.

On a more general level, this publication provides a sample case of domain-specific software engineering and architecture analysis. The same procedure for finding components and defining patterns can surely be applied to other domains as well to provide an overview on the state-of-the-art in system architecture in that domain. If several of such studies exist, comparisons between different domains become possible. Notably, this publication already used several ideas and concepts from general software engineering, e. g. for the discussion on architectural styles in section 7.1.1 and also in some of the patterns. This way it helps to understand which general concepts from software architecture are in actual use in a particular domain. This in turn helps to compare software systems from different domains on a conceptual level for the sake of software engineering research.

Part III.

Domain-specific Item Handling

10. The Core of E-Assessment

Receiving students' input for assessment items and producing grades and feedback for that input is surely the most important duty of e-assessment systems. While the previous parts of this publication tackled organizational and technical aspects that allow many design choices, item handling tackles the core of assessment. Design choices are still possible in that area, but they are now supposed to be primarily determined by the domain of the assessment and not by the process. The purpose of this part is to examine how domain-specific data sources and tools can be used to generate, represent and evaluate assessment items. The focus is thus on the conceptual integration of domain-specific aspects into e-assessment. Pure technical aspects have been discussed in the previous part and can also be found in other places (such as discussions of service-oriented architectures for flexible e-assessment systems [5]).

Different two the first two parts of the publication, the current part is not based on a literature survey, which is due to the enormous mass of research that has been produced in that area in recent decades. Even for the very specific topic of automated grading systems for assessment items on writing program code, more than 100 different systems have been published in recent years [141]. The total amount of literature that examines domain-specific item handling in some way is surely much higher. Hence the current part of the publication will summarize tools and techniques on a rather abstract level to provide a good coverage of the field without running through lengthy lists of details. Instead, chapter 14 is dedicated to detailed case studies from one particular e-assessment system that implements several techniques. Where appropriate, references to other systems or literature reviews will be made throughout all chapters of the current part of the publication.

Assessment items can be divided into different categories depending on the amount of different responses that is possible and that is considered correct. The first category are *closed, convergent* items. In that category, only a limited amount of responses is possible, and consequently only a limited amount is considered correct. Typical examples include multiple choice and multiple response questions, but also items in which elements must be ordered or dragged to a set of predefined positions. In any of these cases, simple mathematical calculations determine the number of possible responses in terms of possible combinations or permutations. Automated evaluation of item responses consequently requires no domain-specific knowledge or techniques in the first place, as all correct responses can be defined during item authoring. An e-assessment system hence only needs to compare an answer with the sample solution and can return the feedback and grade associated with that particular case. However, domain-specific item handling may be necessary when automated generation of item content is used. For example, an item may ask students to order political parties by their result in an election. Both the names

of the elements to be ordered and the expected outcome depends on which election is used during item generation. Consequently, the rule for determining the correctness of a response needs to be adjusted every time new content is generated for the item. Nevertheless, each instance of the item is still a closed, convergent item in which the number of parties determines the number of possible responses and in which exactly one ordering is the correct answer.

The second category are *open, convergent* items. In that category, the number of possible responses is virtually unlimited (with some practical limitation for technical reasons), but the number of correct responses is still limited to a reasonable small number. Typical examples include cloze texts or calculations where a student is requested to enter a single number. In general, each input field in such items allows to type in any sequence of characters or numbers, but only one is considered correct. Tolerance with respect to different representations of an answer (such as ignoring case or allowing both fractions and decimal numbers) may extend the set of accepted answers, but will still keep it small. However, e-assessment systems may need to perform a lot more domain-specific item handling in comparison to closed items. First, the inputs may not only be plain texts or numbers, but any artifact that can be consumed by an electronic system. Examples include mathematical or chemical formulas, images, drawings, and files formatted in domain specific languages like programming or modelling languages. Although it is usually possible to create string representations for any of these contents, determining the correctness of a solution may involve much more domain knowledge than simply comparing the string representations of a student's solution and a sample solution. Domain-specific item handling may also be necessary when generating open, convergent items. In particular, it may depend on the domain what item content is allowed in order to keep the item property of being convergent. For example, a linear function has at most one root, but a sinus function can have an infinite number of roots. Consequently, the following scenario is possible: Students are asked to enter a root of a given function and the item generator may present them a linear function or a linear function combined with a sinus function, but no plain sinus function. The correct answer can be determined directly from the selected function and thus grading and feedback generation just needs to compare the input with the correct answer. However, if the item generator would also use plain sinus functions, there are infinite many correct answers. This requires different techniques for checking the correctness of the input and the item is no longer convergent.

Instead, it belongs to the third category of items, that consists of *open, divergent* items. In that category, both the number of possible responses and the number of correct responses is virtually unlimited. Typical examples include essays and programming exercises. In this category, an item author can only provide a list of properties a response must have in order to be acceptable as a correct solution. Similarly, properties can be defined that trigger specific feedback or result in grade reductions. The process of finding out whether a given solutions holds a certain property can be arbitrarily complex and will require domain-specific tools in any non-trivial case.

The current part of the publication examines the involvement of domain-specific data sources and tools into the item handling in the following way: Domain-specific input tools and data representations are discussed in chapter 11. These aspects are relevant for

any open item types, regardless of being convergent or divergent. Chapter 12 discusses the use of domain-specific data sources and tools for automated item content generation, which is relevant for all three categories of assessment items. Chapter 13 discusses the same for automated grading and feedback generation, which is primarily relevant for divergent assessment items. Notably, the ordering of chapters deliberately does not follow the appearance of the steps in the assessment process, in which item generation comes first and is followed by user input and item evaluation. Instead, the ordering of the chapters is defined from a technical perspective: Input editors and data formats are defined solely based on domain knowledge and are independent of the mechanisms used for item generation or evaluation. Item generation in turn may use domain-specific data formats to express parts of the item content. Finally, item evaluation may do the same and may additionally benefit form information or mechanisms used during item generation.

In all three chapters, techniques, tools and data sources will be reported from literature and compared with each other in order to produce a structured view on how domain-specific elements can be integrated into item handling in e-assessment system. Chapter 14 presents four case studies to wrap up the results from exemplary domain perspectives.

Notably, the current part does not discuss the question whether it is better to formulate a particular question or exercise as an open or closed item. Although this is also a very interesting and important question, it is mainly a matter of instructional design, goals of an exam or exercise and conditions of the target audience. In particular, it is a question that needs to be answered for any kind of assessment, not just for e-assessments. Hence it also does not contribute much to the discussion on how to integrate domain-specific data sources and tools into e-assessments and is thus out of scope for the current publication.

11. Input Editors and Data Formats

As already mentioned earlier, the question of input editors and data formats is primarily important for open questions in which student do not choose from a predefined set of possible answers. Consequently, an item must provide some kind of input fields (at least one) that will naturally only be able to accept input in a specific format. More precisely, each input field will define two aspects: The mode of input and the format in which the input is delivered to the e-assessment system for further processing.

In the general case of domain-independent formats, both aspects may be the same. For example, an input field can accept any sequence of characters from a given encoding and can also pass it to an evaluator component or database. Similarly, an item may accept pictures uploaded as files (thus essentially being a bit string) and send them to other components without changes. However, an item may also contain an input field that accepts free-hand drawings by mouse, gestures or digital pen. Depending on the domain, it can then be appropriate to pass on these drawing as an image or as a formal description of its content. If this decision is deferred to the evaluation step, the input editor and data format stay domain-independent, but if the frontend component displaying the item is already aware of the most appropriate choice, it makes a domain-specific decision.

The decision whether to use a domain-specific input editor or data format is not only a technical design choice, but implies major consequences for the assessment itself: Using a domain-specific format implies the option to reject user input for formal reasons before the actual grading process starts. In particular, it is possible that proper grading is not possible at all, if the input is not understandable for the grading algorithm due to a wrong format. Similarly, providing domain-specific input editors can ensure that the correct format is used but at the same time they may require the user to have specific knowledge on how to use these editors. In that case, users with missing competencies in using the editor may not even be able to make a proper submission at all and will thus receive no grades or detailed feedback. In turn, using a domain-independent format with a generic input editor allows any user to provide any input, including to option to answer correctly by pure guessing. Even if the latter is not seen as a relevant risk, instructional design plays an important role in the decision on which editors are considered sufficient and which are not. Literature reports on cases in which receiving free-hand drawings of mathematical formulas as images was seen as sufficient [215] as well as on cases in which free-hand drawings of chemical formulas as images were not considered appropriate [134]. Notably, some systems may involve multiple ways to submit answers and complex conversions between data formats to allow for a maximum of flexibility in the input [56].

Notably, the discussion of data formats and input modes is a quite old one. It has been mentioned explicitly for the first time not later than 1972: Chemistry students were

asked to turn in their homework in scientific notation on a standard answer sheet that was then transferred to punchcards for automated grading on an IBM 360 Model 50 [55].

11.1. Classes of Data Formats

In general, computers process data in terms of byte sequences. While the individual bytes represent single characters, the ordering of these characters is important to convey the actual information encoded in the data. In the case of e-assessment systems, the most prominent information to be stored is the answer a student provided for an item. The complexity of that information can be very different: In a multiple choice item, the complete answer may be a single character that represents the selected answer option. In an item on business process modelling, the answer may consist of model elements, their connections, textual labels associated with each of them, and layout information for all elements named so far. In the same way, item generation may use item templates to be filled with arbitrary character sequences in a simple case and complex data structures for for the automated creation of large artifacts in other cases. Hence, it may be appropriate for some items to use domain-independent formats to store information of low complexity, while it may be appropriate for other items to use domain-specific formats that are able to encode complex information in a convenient way.

From a technical point of view, defining a data format is equivalent to defining a grammar for a formal language that accepts all byte sequences that adhere to the desired format and rejects all others. Such languages can be either specific for a problem or a domain, or they can be of general use. Moreover, languages may be textual, where any relevant artifact is created directly in that format, or they may be graphical with one or more additional textual representation schemas. Examples for all these cases and their characteristics will be discussed in the following subsections.

11.1.1. Plain Text Format

The most common data format that can be found in many e-assessment systems is plain text. In particular, it can be used with fill-in-the-gap items and any kind of essay tasks. The data format contains no information about the domain of the item and thus the answer for two completely unrelated items from different domains may look the same. Any answer will be subject to grading, as the format itself does not allow to make any distinctions between a meaningful answer and a meaningless one. The only thing that can be ruled out safely is an empty answer.

Plain text can also be used during item generation e. g. to store contents to be inserted into placeholders in an item template. It allows for great flexibility as it can contain virtually any content, but implies limited control about the generated item. For example, the character sequence “1/2” may have the same meaning as “2/4” in the domain of mathematics, but a different meaning if it is meant to be a house number. Thus any domain-specific logic that allows for validations, simplifications or unifications needs to be encoded in the item generator that knows the domain, as it is not visible from the format.

11.1.2. General Purpose Data Formats

Another common feature of e-assessment systems is the use of general purpose data formats for parts of the items and also for encoding answers. One available option are various file formats for encoding images or drawings, like JPEG, PNG or SVG. These can obviously be used in many contexts to include images into items during item generation and they can also be used to submit answer as scans of free-hand drawings on paper. Similarly, a system may request a specific file format for textual input as an alternative to plain text [316]. Proprietary formats may fall into the same group of data formats, even if they were designed with a specific domain in mind. For example, a format that was designed for free-body diagrams in mechanics teaching [299] is most likely usable also for many other kinds of drawings or sketches as well.

Proprietary formats for domain-independent closed items are also an example for general purpose data formats. For example, the choice made by a student in an multiple response item can be encoded as a sequence of 0 and 1. While the resulting character sequence is plain text from a technical point of view, it is based on a simple formal language. That language restricts the choice of possible characters in the answer and also its length. Similarly, a proprietary format can be used to encode choices made during the item generation process.

Different to plain text, general purpose data formats allow to reject answers if they are in the wrong format. However, that decision is in no way related to the domain of the item and thus equivalent to the decision on whether a plain text is empty or not. For example, even a completely black or white image will be accepted for grading, as long as it is in the correct file format. Hence, again the very same content can be submitted to completely different items from different domains and will be handed over to the respective evaluator components. The only way to ensure that the answer is meaningful is to use input interfaces that create some restrictions like the checkboxes mentioned above. In that case, the answer adheres to a specific format due to the way it was created. In fact, this is the usual way of using general purpose data formats, as students are usually not asked to write the correct file format by hand or encode there answers themselves. Instead, an appropriate input editor is used or students are asked to upload files that have been created with a suitable tool.

Similarly to answer grading, contents produced during item generation can only be rejected in an automated validation for formal reasons. Since item generators can use standard libraries in many cases (e. g. for image generation [217]), rejecting created content for formal reasons is a largely hypothetical case.

11.1.3. Textual Domain-specific Languages

Many domains make use of domain-specific languages and in some cases it may even be the aim of an assessment item to expose students to that particular language. A typical example are assessment systems for programming, where students are expected to submit program code [141]. In that case, answers that do not adhere to the expected format are not necessarily rejected. Instead, feedback can be given on syntactical errors, which is in

fact a very common class of feedback for programming items. However, there is no clear border between meaningful answers that contain a large amount of syntax errors and a meaningless answer in the wrong format. Hence, any answer must be processed by the domain-specific format check and it must be decided on the basis of that result whether feedback is created or the answer is rejected.

Also other systems that do not specifically aim for training students in a particular language make use of domain-specific languages to capture complex input in a convenient way. Most prominent is the use of the LaTeX language in e-assessment systems for mathematics [236]. Instead of implementing domain-specific logic in the evaluation component to find out that "1/2" might be a fraction with numerator 1 and denominator 2, it is left to the student to write that fraction in LaTeX code and thus directly mark the numerator and denominator. That is a method that is also applicable to larger and more complex artifacts. Similar to the case of programming languages, it allows to reject answers that contain no LaTeX code and it allows to give precise feedback on syntax errors before it comes to actually grading the answer. Similar is possible for other formats for mathematical input that follow the syntax of mathematical software tools [214].

Since domain-specific languages allow for feedback on syntax errors, the use of format-specific input editors has important consequences. The editor may be non-intrusive, like most IDEs for programming are. In that case, it marks syntax violations and hence gives some kind of feedback, but it does not prevent uses from saving their wrong input. Consequently, syntax errors can also be detected during the grading process, which allows to trigger respective feedback. However, the input editor may also be intrusive and thus hinder users in creating artifacts that violate the grammar of the domain-specific language. For example, an input editor may try to render the LaTeX input and reject any input that is not in correct syntax. In that case, any answer that reaches the evaluator component is syntactically correct. While that may be beneficial by simplifying the grading process, it may also be a drawback as no feedback can be produced for a particular class of errors.

Domain-specific languages can also be used to store data during item generation. In that case, standard tools for that language can be used to validate the generated constructs or to perform simplifications and unifications, like doing some pretty-printing for generated program code fragments or flagging LaTeX commands with empty mandatory parameters.

11.1.4. Domain-specific Languages with Additional Representation Schemas

Besides textual languages, many domains make use of graphical languages. Since parsers for graphical representations are rare, additional textual representation schemas for most of these languages have been defined. The graphical representation encodes information in a human readable way that is often convenient to read and edit by using the concrete syntax of the language. At the same time, the textual representation encodes the same information using the abstract syntax of the language within a different representation schema, including extra information like layout positions and alike.

Probably the most common domain-specific language of that kind is the language used in mathematics. It uses various text markups and special symbols (like indices, superscripts, or the sum symbol with all its various annotations) as well as artifacts

that require position information (like vectors and matrices). Although many systems use the LaTeX language to encode mathematical formulas and expressions as discussed above, that language does not qualify as an additional representation schema for the mathematical language. Admittedly, LaTeX provides some commands that are named and formed directly after the mathematical constructs they are used for (such as the LaTeX command for fraction), but in general LaTeX is just a markup language that reflects the optical properties of mathematical formulas and expressions but not their semantics. Instead, domain-specific languages such as OpenMath[1] and MathML[2] can be used in e-assessment system [53, 248, 285]. These languages are based on XML and are designed specifically to capture the abstract syntax of mathematical formulas and expressions in a machine readable way. Similarly, OpenChem and ChemML[3] can be used for chemical formulas [216], XMI for UML models [206] and Music XML[4] for musical scores. For the latter, VexTab[5] is an alternative that is not based on XML.

Different to textual domain-specific languages, syntax errors can occur in languages with additional representation schemas on two levels: First, there could be a violation of the grammar of the representation schema (e. g. XML). If that occurs in an answer, it can be ruled out as invalid. Second, there could be a violation of the grammar of the actual language (e. g. UML). If that occurs in an answer, it can be considered in the feedback. Thus domain-specific languages with additional representation schemas allow for a better distinction on when to reject an answer for formal reasons and when to give feedback. That is also compatible with format-specific input editors, as these can be designed in a way that they ensure the correct use of the representation schema, but allow to make syntax errors in the domain-specific language. However, that benefit can only rarely be used to full extent for two reasons: First, many universal input editors that are not specifically designed for educational purposes prevent users from making at least a large amount of syntax errors. Moreover, many standardized representation schemas are only able to represent syntactically correct artifacts. Hence, the creation of editors and representation schemas that explicitly allow to create and store artifacts with errors is sometimes an explicit requirement for e-assessment systems to allow teachers to inspect those errors and use them for teaching [292].

Domain-specific languages with additional representation schemas can also be used for item generation. Similar to pure textual domain-specific languages they allow for validation, simplification and unification of the generated artifacts without the need to implement specific logic for that in the item generator. Instead, standard tools for the particular language can be used that are able to e. g. turn the fraction $\frac{2}{4}$ into $\frac{1}{2}$ or to find a neat layout for a generated UML diagram.

[1] https://www.openmath.org/
[2] https://www.w3.org/Math/
[3] http://cml.sourceforge.net/
[4] https://www.musicxml.com/
[5] http://vexflow.com/vextab/

Format	Formal check	Format-specific input editors	Domain-specific operations during item generation	Usage examples
Plain text	not possible	not possible	not possible	Essays
General purpose language	reject	typically used to enforce format	only format validation	Images, MC indices
Textual domain-specific language	reject or feedback	can help but may suppress feedback	possible	Programming
Domain-specific language with additional representation schema	reject or feedback	can help and allow feedback	possible	Mathematics, Conceptual modelling

Table 11.1.: Summary of key characteristics and capabilities of different data formats used in e-assessment systems.

11.1.5. Summary

As a result from analyzing the examples of domain-specific and domain-independent formats in the previous sections, we can conclude the following definition: A data format is domain-specific, if assumptions about the format or content of an item or submission can be made that go beyond technical specifications (like file format or character encoding) and that may rule out answers as invalid before it comes to answer evaluation. When using a domain-specific data format, an invalid submission is not considered a wrong submission with respect to the goal of the assessment and for the purpose of giving elaborate feedback. The use of domain-specific formats allows for validations, simplifications and unifications as an automated step during item generation. In contrast to that, a data format is domain-independent, if there are at most assumptions about the technical specification of the format or content of the item or submission. All submissions are evaluated and problems in understanding a submission are causing it to be considered wrong, potentially triggering elaborated feedback.

The different possibilities for formal checks, use of format-specific input editors and domain-specific operations during item generation are summarized in table 11.1. The choice of examples stresses that the two top lines in the table contain domain-independent formats, since the refer to item and content types rather than domains. In turn, the

examples in the two bottom lines indeed refer to domains and thus point out that the respective formats are domain-specific.

11.2. Classes of Input Editors

The discussion of data formats in the previous section directly triggered a related discussion on input editors. Whenever a data format other than plain text is used, input editors can help students to make proper input. At the same time, input editors that enforce certain properties of the format can simplify the development of evaluation components, since less checks for correct formats are necessary. As already mentioned above, that may in turn hide student errors from the evaluator component and thus limit the capabilities to give feedback. In turn, domain-specific editors may produce feedback on syntactical errors on their own without involving any other components. If the grading is split up that way as a deliberate design choice, that is a concrete example of the *mixed processing* pattern that was discussed in section 7.3.4 in part II.

Closely related to the possibility to produce feedback are the user's competencies that are required to use the input editor. For example, a simple drag-and-drop editor may be usable for anyone with reasonable skills in using a computer, but it is only able to provide information on the positions of dragged elements in a general purpose data format. In turn, a sophisticated editor may produce a formally correct diagram of a conceptual model in some standardized representation schema, but at the same time it requires the user to know at least some rules on how to combine available elements.

For most standardized data formats there are also editor components available that can be integrated into an e-assessment system. As already mentioned above, that has serious consequences on the feedback. Most available input editors are not designed with educational purposes in mind. They thus may simply prevent users from making some types of errors without giving elaborate feedback on why some editing step is not possible. While that can be helpful during practice and training, it may be limiting in the context of assessment. With such editors in use, it cannot be distinguished whether students did not make a particular error because of their competencies or because the editor did not allow to make that error.

If no appropriate input editors are available, e-assessment systems need to implement own components for that purpose. These editors may include extra features for educational purposes that go beyond producing output in the desired data format. In particular, input editors may store log information on how the editor was used to include it into the grading process for a deeper analysis [28]. In turn, e-assessment systems may want to offer specific editors for some item types and thus also need to introduce an appropriate data format. Since these item types may or may not be domain-specific, the resulting data formats also may or may not be domain-specific.

Notably, input editors are not necessarily integrated into the student interface. Also external tools qualify as input editors, even if students need to export their contents from their as files and upload them into the e-assessment system. However, these external tools create no separate class of input editors and the following sections make no distinction on

whether a particular editor is offered as an integrated feature within the student interface or as an external tool. Similar is true for the integration of input editors into other parts of an e-assessment system, i. e. into teacher interfaces or authoring tools. Although the following sections are phrased from the student perspective, similar considerations apply for teachers and item authors.

11.2.1. Examples of Domain-independent Input Editors

One typical class of domain-independent input editors are plain keyboard based text input fields, such as in fill-in-the-gaps or (short) essay items. They are often used in conjunction with plain text data format and thus offer no support for avoiding syntactical errors or giving feedback on any aspect of the content. They can also be used in conjunction with general purpose data formats and offer feedback or comfort features related to that format such as syntax highlighting or spell checking. It can be expected that any student with reasonable competencies in using a computer is able to perform meaningful interactions with these kinds of input editors and it can also be expected that they are usable on a very large range of devices. In turn, students will at most get some convenient comfort in doing their inputs, but they will not be able to learn anything about the domain of the item just by using the input editor and observing how it handles their input.

The same is true for pointer-based interactions as another common class of domain-independent input editors. They occur in cases where students have to point and click on select boxes or where they have to drag and drop elements into place. Also editors in which free-hand drawings can be made that are stored in a general purpose data format as raster or vector images belong to that category. Even sophisticated interfaces that allow students to pan and zoom images on which they have to place markers belong to that category, since the editors only store plain coordinates. In fact, the very same technical framework can be used in completely different domains [134].

11.2.2. Examples of Domain-specific Input Editors

Probably the most prominent class of domain-specific input editors is the class of editors for line-oriented graphical notations. They are typically used for mathematical formulas [236], but examples also exist for other domains like chemical formulas [216] or musical scores [203]. Different technical solutions exist to allow for the correct placement of the various formula elements, including the definition of control keys on the keyboard (such as automatically turning "x^2" to x^2 and "x_2" to x_2), the definition of control sequences (such as selecting a note by mouse click and then use "Shift+Arrow Keys" to move it on the lines) or the provision of template palettes as shown in figure 11.1. A common characteristic of all these features is that they support students in making their inputs by rejecting constructs that are not possible in the respective domain-specific language. In cases in which plain rejection is not possible (e. g. because partially compete constructs must be possible due to technical reasons), these editors can at least give some feedback on their own on the formal correctness of the input without involving any other component. Different to plain text input fields, these editors require a bit more

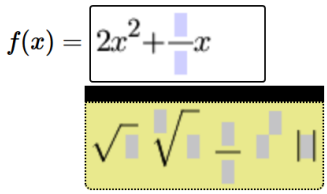


Figure 11.1.: Sample UI of an input field with a formula editor. The palette provides elements for some typical mathematical notations for roots, exponents, fractions, and alike. They can be added to the input field by mouse clicks.

competencies to be used correctly. In turn, even students with no prior knowledge will be able to learn at least some bits of information on the correct usage of the domain-specific notation just by using the editor and observing which input is possible and which is not. That immediately raises the question on whether using such editors is necessary or not. The answer depends on the expected data format. If input can be stored in a general purpose data format, a domain-specific editor is optional. Instead, a general purpose graphical editor that allows students to create e. g. formulas as drawings is sufficient [215]. However, if a domain-specific data format must be used, but students should still have the opportunity to see the usual graphical representation, an editor that is able to handle two representation schemas at once is mandatory.

Editors for textual domain-specific languages are another class of domain-specific input editors. They are common in the domain of computer science for items on writing program code, but can also appear in other contexts. Similar to domain-independent textual editors they may offer features like syntax highlighting or autocomplete based in the language grammar. However, they have to be counted as domain-specific input editors if these features are used in cases in which knowing the domain-specific language is part of the required competencies to answer an assessment item. In particular, students without any prior knowledge in the domain of the item are again able to learn about the domain-specific language just by observing how the editor understands their input. Hence, also spell checkers for plain text input can be counted here, if the domain of the item is language training and the essay they write is on an arbitrary topic. However, the very same editor is not considered domain-specific, if the domain of the item is the topic of the essay, while students are expected to know the grammar of the language they use anyway. Different to the class of editors for line-oriented graphical notations discussed above, editors for textual domain-specific languages can be considered an optional comfort feature in any case, as there is only one representation schema that is written directly by the students.

A third major class of domain-specific input editors covers any kind of graphical notations that is not considered among the line-oriented ones. Similar to those, it is also intended to be used with domain-specific languages with additional representation schema. However, control keys and template palettes are not enough for graphical languages in which students are free to arrange elements in an arbitrary layout. Instead, editors usually offer mouse control to place and resize elements. Additionally, they may accept keyboard

input to add texts to diagrams and to make layout modifications via control keys or sequences. Using such editors usually requires not only knowledge in the domain-specific language they support, but often also in the features of the actual editor, as different editors may offer very different ways to achieve the same result. In turn, these editors may contain sophisticated features that reject illegal constructs or give feedback on how to improve them to conform to the grammar of the respective domain-specific language.

11.2.3. Summary

Similar to the discussion on data formats, also the discussion on input editors allows to conclude some kind of definition for the different input mechanisms: The input mechanisms to an item are domain-specific, if at least one element to interact with in the item presentation requires competencies that go beyond general competencies in using computer interfaces. People with insufficient competencies in the particular domain of the assessment may not be able to interact with the item in a meaningful way. In turn, a domain-specific input editor can enforce domain-specific data formats and can give feedback on syntactical errors on its own. In contrast to that, input mechanisms to an item are domain-independent, if the elements to interact with in the item presentation require not more than general competencies in using computer interfaces. Even people with no competencies in the particular domain of the assessment can create a correct submission by guessing without learning about any syntactical rules. In turn, a domain-independent input editor can at most enforce general purpose data formats and cannot provide any meaningful feedback on syntactical errors.

The relations between the different classes of input editors and the use of domain-specific or domain-independent data formats are provided in table 11.2 in conjunction with key characteristics and usage examples.

Class of editors	Typical data format	Integrated or external tool	Exemplary means of support and integrated feedback	Usage examples
Domain-independent keyboard based	Plain text	Typically internal	Spell checking	Fill-in-the-gaps, Essays
Domain-independent pointer based	General purpose	Typically internal	none	Multiple choice, Drawings
Domain-specific textual notation	Textual domain-specific language	Both	Syntax highlighting, spell checking, autocomplete	Program code
Domain-specific line-oriented graphical notations	General purpose or DSL with add. representation schema	Typically internal	Templates, rejection of illegal input	Formulas, Musical scores
Domain-specific arbitrary graphical notations	General purpose or DSL with add. representation schema	Both	Rejection of illegal input	Diagrams

Table 11.2.: Summary of key characteristics of different classes of input editors used in e-assessment systems. (DSL = Domain-specific language)

12. Automated Item Generation

A major challenge in preparing an assessment is the generation of a set of assessment items that is sufficiently large and where each item has a sufficient quality. Section 3.1.2 in part I already discussed that challenge from a process point of view and section 7.4.2 in part II named several technical patterns for item generation on a rather abstract level. The current chapter now tackles the question how domain-specific data sources and tools can be integrated into the process of automated item generation. While most examples throughout the chapter use classical examples from automatic item generation based on item models [101, 100], the general techniques of integration can also be applied to other content construction approaches like the cognitive design approach. Differences between these different approaches exist from the psychological perspective of test and item design: The item model approach requires a fairly low cognitive foundation and can thus be used for item generation relatively quickly. In contrast to that, the cognitive design approach also quantifies the level and the source of cognitive complexity in an item and thus makes construct validity more explicit [82]. The current chapter acknowledges these discussions and the many results on psychometric properties and item quality (e. g. [15, 123, 16, 17, 21]), but takes a technical perspective on automated item generation. That focus does not imply any conceptual drawbacks, since research has shown that automated item generation can produce items in the same quality as manual item generation [220]. Further details on the psychometric use of assessment items are discussed to some extent in part IV of the current publication.

There has been much research on automatic item generation in recent years. In particular, there is a lot of results on automated item generation for multiple choice questions. That is not surprising, as the psychometric properties of that item type are well understood and different models (as mentioned above) can be applied. Automated item generation for multiple choice items appears in many domains, including but not limited to medicine (e. g. [4, 104, 157, 102, 252]), language training (e. g. [286]), math (e. g. [295]), biology (e. g. [8, 293]), and computer science (e. g. [137, 282]).

However, there are also many cases of automated item generation for other item types across various domain, including but not limited to math (e. g. [113, 186]), computer science (e. g. [251, 54]), and electrical engineering (e. g. [258, 174]). Automated item generation is thus suitable for all kinds of items and not limited to closed or convergent ones.

The aim of the current chapter is to summarize and organize knowledge about the technical process of domain-specific item generation in a more abstract way than on the level of individual cases for a single domain. At the same time, it aims to stay as close to the practical problems of generation algorithms as possible to come up with a description in which the need for the integration of domain-specific tools or data-sources

is still visible. Hence, any description that considers each case of domain-specific item generation as an individual case that can be solved by a straight-forward, monolithic item generation algorithm is too narrow, while any description that handles the creation of item content as an atomic step within a larger process is too abstract.

12.1. An Anatomy of Assessment Items

The basic idea in all cases of item generation is that an item consists of one or more structural elements (such as words, sentences, numbers, pictures and anything alike) and that there is at least one dependency either between these elements or to some external source of knowledge. For example, an item may consist of a text and a question that asks for one of the facts contained in the text. There are surely several texts that can be used in such an item and there are also several questions that can be asked. However, the item is only meaningful if the question can be answered by reading the text and hence there is a dependency between those two elements. Even in a simpler case, in which the item contains just a question for some fact a student is supposed to know, there is a dependency between the question and some knowledge base. Even if that knowledge base is external, the dependency is also encoded in the item, if a grading rule is included in the item that allows to decide on the correctness of the answer. That rule may either make an explicit reference to the external knowledge base, or it may contain a copy of the relevant piece of information from there. Details on how such grading rules can be defined are discussed in chapter 13. Regardless of the actual mechanism, item generation has to make sure that the dependency is established correctly and is encoded correctly within the item. That is true even in cases in which there is neither a direct dependency between the element nor an explicit dependency to a knowledge base: In an item asking for the sum of two integer numbers, these two numbers are completely independent. There is also no knowledge base where one can look up some numbers. Nevertheless, the answer obviously depends on the numbers and hence any grading rule must encode a relationship between these numbers and the answer.

Notably, there is no unified or standardized way on how to encode the elements of an item and their dependencies in a technical item description. Standardized formats like QTI [131] not only allow to define fixed items, but also allow to use templates and placeholders and thus encode variable elements. Besides that, there are also system-specific formats like for example the ones in the e-assessment systems JACK [154] and PILOT [39] or in the SARAC framework [165].

For the specific case of multiple-choice items, Gierl et al. [103] created at least a precise taxonomy of the different elements of an item and their relationships. In that taxonomy, a multiple-choice item consists of a stem and options. A stem can consist of either (1) independent element(s), (2) dependent element(s), (3) mixed independent/dependent element(s), or (4) fixed element(s). Options are either (1) randomly selected, (2) constrained, or (3) fixed. Optionally, an item may also include auxiliary information that may be either (1) dependent or (2) fixed. The taxonomy paper also reports on a study in item generation in which 331,371 items where generated. The vast majority

of 202,860 items falls into the category of dependent stem elements and constrained options. Additionally, the category of mixed stem elements and constrained options accounts for 122,880 elements. Hence, the details on how elements are dependent and constraint by each other deserve a closer observation. Notably, the taxonomy is designed for multiple-choice items only and thus does not explicitly cover the fact that in a multiple-response item two different properties can be true for answer options, e. g. two constrained correct answers and three random wrong answers within the same item. In a similar way, the taxonomy can be extended for any kind of closed item. In turn, the classification of the options can be skipped for open items. Instead, these items may contain grading rules which may show the same four properties than the item stem.

The mere fact that most of the items reported in the taxonomy paper contain constrained options does not imply that these constraints are domain-dependent. In the case of a mathematical item it is quite obvious that a generation algorithm must be familiar with the domain of mathematics to be able to generate a correct grading rule. In the example given above, the generation algorithm must be able to perform an addition on the two independent numbers in the question to create the sample solution for a grading rule. However, not all cases are that clear and the dependency on a domain is not always given. In particular, there are several approaches to automatic question generation that make use of algorithms for natural language processing [153]. These algorithms are able to generate meaningful questions (including a correct sample answer) for virtually any text from any domain. In that case, the generation process can be seen as domain-independent, as it basically only transforms input into output without knowing anything about the semantics of the input. In [4] it is explicitly mentioned that their approach needs no prior knowledge in semantic relations. Many approaches to question generation via natural language processing make use of machine learning learning approaches. Some of these approaches might benefit from models that are trained for a particular domain, so that the generation is no longer domain-independent. However, even in these cases that actual algorithm is still domain-independent and it will also work with domain-independent data. The only difference is, that the quality of output will be lower.

Different to that, approaches that use ontologies instead of text as input are really domain-dependent. Although the algorithms included in the process may still make simple transformation, the input itself is domain-dependent in the way that ontologies capture the semantics of the relationships between entities in a particular domain. While the grammatical structure of two texts on different topics may be identical, the structure of two ontologies from different domains is not. In particular, an approach that is designed to work on a particular ontology may fail to produce any result on a different one. Different to the machine learning approaches named above, the input decides not only about the output quality, but it decides whether there is meaningful output at all.

Hence the following sections will deal with cases in which domain-specific knowledge is encoded in the generation algorithms (like in a mathematical item) as well as with cases in which domain-specific data sources like ontologies are used as an input. Moreover, also cases in which plain natural language generation algorithms are used are considered for the specific case of item from the domain of language training. For the algorithmic approaches,

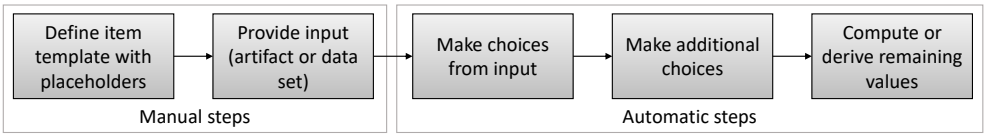


Figure 12.1.: The process for automatic domain-specific item generation.

two sub-sections will distinguish between approaches that rely on domain-specific general purpose software that is re-used for item generation and specific algorithms that are designed exclusively (or at least primarily) for the purpose of item generation.

12.2. Item Generation Process

The remainder of the current chapter considers an abstract process for automatic domain-specific item generation. That process consists of five steps in two phases (see Fig. 12.1): A manual phase is dedicated to the item model creation and consists of two steps, while an automated phase performs the actual item generation and consists of three steps. The process assumes that a cognitive model structure has been created beforehand, so that the structure of the respective domain as well as the relevant facts and data sources are known. Actual approaches to automatic item generation may use variants of the process in several ways. First, some steps are optional and can be omitted if they are not necessary for a particular class of items or a domain. Second, one conceptual step may be broken into two or more sub-steps for technical reasons and the order of such steps may be exchangeable. Examples of such variations will be discussed in more detail in the case studies in chapter 14.

In the first step, an item template is designed by creating text skeletons for item stem and answer options that may contain an arbitrary number of placeholders. In particular, answer options may consist of placeholders only, while the item stem typically contains at least some fixed pieces of text. Answer options are not necessarily handled in an uniform way, so that there might be a set of answer options with placeholders and another set of fixed answer options.

In the second step, a data source or artifact to be used during the generation process is provided by the item author. That can be done either by providing a suitable document directly or by providing a reference such as e. g. a repository URL and a SPARQL-query to retrieve data from that repository. In any case, data must be provided in a structured and machine-readable format, as it is analyzed during the generation process. Hence, the artifact or data set provided is not exactly the same as the "auxiliary information" named in the taxonomy by Gierl et al.. Notably, up to here nothing domain-specific has been done in the process.

The third step is then the first automated step in the generation process. It is an optional step that makes a (possibly constraint) random choice from the provided artifact or data set to choose the contents that are actually included in the item. For example, the provided data set contains some statistical data for all European countries, so that

the generation algorithm can randomly choose two or three of those to be used in an actual item instance. Constraints may be formulated by the item author to ensure that the selected elements of the data set are different in some way. Both the way how to select data from the data set and the potential constraints are domain-specific.

In the fourth step additional random parameters are chosen by the generation algorithm, which is again an optional step. The choice for each parameter may be constraint in some way by the respective domain or by the data selected in the previous step (or both). However, the choice may also refer to other elements of the item template. For example, the item template may define that there are four groups of statements to be used as answer options, where each group contains several templates with placeholders. The fourth step can then randomly chose one statement from each group to be included in the actual item instance.

In the fifth step, the values for all placeholders in the templates are computed, if they are not filled with values produced in the previous steps. This may involve arbitrarily complex domain-specific operations that analyse the given artifact or data set and make any computations that are necessary to create values, texts, or even graphics.

Since the process is meant to describe an abstract schema for automatic item generation, it makes no assumptions on when the steps from the second phase are actually performed. In particular, the process is valid for "offline" generation where a complete item instance is generated within an item generator software and stored in an item bank for later use as well as for "online" generation where an item is created on demand within an e-assessment system. Any mixture of these two approaches is possible as well. That also includes cases in which input to the item generation process is derived from previous inputs from students. Since the item generator component consequently deals with students' input similar to an evaluator component, there are some aspects that can be discussed in the current chapter and in the following one on answer evaluation. To avoid duplicate discussions, appropriate references will be made throughout both chapters.

12.3. Item Generation Techniques

Each of the steps three to five in the generation process bears the possibility to perform domain-specific activities or to work on domain-specific data. The different possible ways of data handling and computation are discussed in the following sub-sections.

12.3.1. Generation with Ontologies

Items that ask for factual knowledge form a large class of items that can be used in many different assessment scenarios. Moreover, factual knowledge can be assessed in many different item types, including closed item types as well as open, convergent item types. Consequently, there is a high demand for such items. Several approaches to item generation for these types of items make use of ontologies, because these offer a way to encode facts from virtually any domain in a machine-readable way. These ontologies are then the input artifacts used in step two of the item generation process and data from the ontologies is selected in step three.

In the easiest case, there is only the need to retrieve some single facts from the ontology with quite simple queries to complete all elements of the item template [6, 88, 317]. The domain-specific dependency between these elements is hence encoded completely in the data source and no additional choices or computations need to be made in the subsequent steps four and five. The quality of information in the data source and the way the data is retrieved has nevertheless an important impact on the quality of the generated item. For example, a classical multiple-choice item may need one correct answer option and three distractors. These distractors must be carefully chosen to ensure a high item quality and thus the domain-specific relation between the correct answer and the potential distractors must be considered. In particular, it might not be sufficient to rely on general properties of ontologies, such as using random entities that all share a common parent class [8].

This way of item generation is not only suitable for closed items, where both the correct answer option(s) and some distractors are taken from the ontology. Instead, it can also be employed to generate open items in structured domains like SQL databases [125]. In that case, the ontology is itself subject of the item and a generated query problem is given in natural language in the item, while a sample solution in terms of a SQL query can be generated automatically.

Notably, an ontology may contain references to other data sources and provide semantic information on the contents that are available there. For example, an image repository can be used to generate items for medical training, where an image used as part of the item stem is related to information on diseases in an ontology that provides answer options and distractors [252]. Similarly, an ontology can provide information on chemical compounds, while images showing the respective structures are located in a separate repository [270]. Nevertheless, the key to domain-specific information in that case is the ontology, while the separate images only serve as auxiliary information. They may contain relevant and necessary information for answering the item correctly, but they are not central to domain-specific item generation.

Automatic item generation from ontologies is not limited to factual knowledge. The fact that ontologies are structured along domain-specific relationships between elements can be exploited to create items that ask students to analyse and classify objects or statements. For example, entities that belong to different concepts can be retrieved from an ontology (possibly enriched by images or alike from an additional data source) and students are asked to classify these entities according to a given criterion (e. g. classification of ancient pottery in archaeology [273]). While the necessary mechanisms for item generation are similar to the ones discussed above, such items require the competencies to analyse entities or apply classification methods and do thus target higher competency levels than items that require to remember facts.

Item authors can use a wide range of different data sources for automatic item generation with ontologies. There are general purpose data sets available such as Wikidata[1] and DBpedia[2] that provide concepts from many different domains and that are extensible for virtually any facts. There are also many domain-specific data sources that provide

[1] https://www.wikidata.org/
[2] https://wiki.dbpedia.org/

structured data based on fine-grained ontologies for a particular domain, such as the British Museum[3] provides for data from their collection. Finally, there are approaches that provide an ontology-based access to other data sources like the Sophox-project[4] does for geographical data from the OpenStreetMap-project. In any case, item authors must make themselves familiar with the ontologies. In particular, not all data sources are appropriate for all kinds of items [89]. Moreover, item authors must potentially explore the possibilities to make combined queries from more than one data source. Consequently, there will not only be domain-specific knowledge encoded in the data sources, but there will also be domain-specific knowledge in the queries use to access these data sources in step three of the item generation process.

12.3.2. Generation with other Semantic Sources

Since ontologies are not the only means to capture information on semantics in a structured way, other approaches have also been explored for item generation. The linguistic concept of Frame Semantics can be used to create questions in natural language without using fixed templates [66]. The reasoning behind that idea is the fact that e. g. mathematical problems have a different level of difficulty depending on the way they are phrased. While fixed templates require at least a uniform sentence structure for all possible instance and only allow to exchange words and numbers, Frame Semantics allows to generate sentences with different structures that have the same semantics. Each semantic frame in that theory provides an encoding of a domain-specific set of entities, operations and the relations between them, along with typical words used to express statements about these elements in natural language. Hence, a domain-specific item generator is able to express e. g. the same mathematical problem in the wording of different domains. In fact, that can be used in two ways: First, it allows to use the same semantic frame (e. g. the domain of time-distance-travel-problems) to find wordings for problems from different domains (such as mathematics or geography). Second, it allows to express problems from on domain (e. g. mathematics) in different semantic frames (such as medicine or economics). In both cases, the approach is applied in step five of the item generation process, after all relevant contents have been derived from other sources (step three) or have been chosen randomly (step four).

12.3.3. Generation with Domain-Specific General Purpose Software

Whether or not a domain-specific data source has been provided in step two of the process to choose data from it in step three, steps four and five are concerned with additional (constraint) choices and computations, respectively. One option to automated these steps is to integrate software components that are capable of performing some domain-specific operations, but that are not specifically designed for item generation. The most prominent example of such systems are computer algebra systems (CAS), that

[3] https://collection.britishmuseum.org/sparql
[4] https://wiki.openstreetmap.org/wiki/Sophox

are of general use (not only) for teaching mathematics [160] and that can be found in many e-assessment systems (e. g. [241, 107, 36, 248]).

A CAS typically is able to perform complex mathematical operations that go far beyond simple arithmetic operations that are provided by most programming languages. They can thus be used for example to create the correct derivation for a random polynomial to be used as an answer option in a multiple choice item or as a sample solution in an open item type. Notably, in the latter case a CAS can also be used to evaluate the answer, which is why it will also be mentioned in chapter 13 below.

Similar to computer algebra systems are natural language generation (NLG) systems, that are able to create grammatically correct sentences for a particular language from some input parameters. Different to the general use of natural language generation techniques used for domain-independent item generation from input texts, NLG systems can be used for example in the context of language training to create sentences with specific grammatical properties [216].

In structured domains like computer science, appropriate general purpose software can be used in very specific cases. For example, constraint solvers that are typically used for formal methods in software verification can be used to generate sample instances of object diagrams for given class models and thus create content for items on software modeling [137].

12.3.4. Generation with Generic or Domain-Specific Algorithms

Whenever general purpose software is not available for a particular domain or does not offer the necessary functions, specific solutions must be implemented. That typically results in monolithic item generators that may allow for some kind of parameterization, but that are in general specific for a particular item type in a particular domain. Both the complexity of the involved algorithms and their actual duties depends very much on the individual case.

In simple cases, the algorithms just reflect the actual concepts an item is about. For example, answer options in a multiple choice item on a Caesar cipher can be generated exactly by applying the cipher algorithm to a random input sentence [213]. No additional implementation is necessary in that case and thus the domain-specific part of the item generation just happens in step five of the generation process. A similar situation for step three of the process can be found in cases in which the input provided in step two requires a domain-specific and item-specific analysis. For example, UML models can be analysed automatically to identify the elements contained in these models and subsequently chose some of these elements to include them in an item template [217]. In that case, all relevant operations are encoded in a domain-specific algorithm for step three of the process, while the other steps can work in the same way as they would do with simpler input.

Slightly more complex are cases in which some variance must be included, since there is not a single correct solution. In that case, the item generator must generate all possible solutions in step five of the process. For example, an item on advanced algorithms on data structures can ask students to sort elements in an array following a specific algorithm.

While the final result is similar in all cases, the intermediate steps may depend on the chosen variant of the algorithm [148]. In that case, the item generator must simulate all variants to allow the grading component to compare a student's solution with any of them. A different option is to generate only one of several correct answers automatically and defer the identification of syntactically different but semantically equivalent solutions to the answer evaluation [222].

Even more complex are cases in which an algorithm produces more output than just a set of uniform elements (such as encrypted sentences, model elements or possible solutions in the examples discussed above). For example, specific algorithms for electrical engineering can be used to generate circuit diagrams and also do all computations that are necessary to compute some properties of the circuits [257, 174, 256]. Similarly, a tree-based algorithm can be used in an generator for computer science that generates an expression and its correct interpretation [149]. Such algorithms may thus span all three automated steps of the generation process: They may take some input parameters that constrain the design of the resulting circuit; they may make some additional, subsequent choices; and they do domain-specific calculations to come up with the correct results to be used in answer options for closed item variants or grading rules for open item variants. Obviously, the process can be simplified by skipping the last step if only a problem is generated but no sample solution for automated grading [251].

There are also cases in which less steps are involved, but the necessary operations are more complex. For example, the generation of programming exercises including some code templates requires little input and thus no complex algorithms for step three and four. However, automatic generation of the necessary test cases that are involved in the grading process is a highly domain-specific task. It can be automated in the way that an item author provides an abstract test specification (i. e. possible inputs and expected outputs), while the actual test code is generated automatically [54]. The latter can even happen in different programming languages, so that several variants of an item can be generated from the same input.

Even more domain-specific knowledge is involved in the generation algorithms if the input also needs domain-specific analysis. In the case of programming exercises, an item generator can analyse a piece of source code as input, detect appropriate sections of that code to be replaced, compute meaningful replacements for these sections and finally compare the output of the modified code (based on random but meaningful call parameters) with the original output [282]. Based on these computations, a multiple choice item can be generated that asks which replacements do not change the original behaviour. That particular item will be discussed in more detail in section 14.3.2 below. It involves domain-specific operations in step three to analyse the input, in step four to make meaningful choices on random replacements and in step five to compute the correctness for each answer option.

12.4. Summary

The discussion of the item generation process and the actual techniques that can be used in each of the process steps provide a good insight into the differences of domain-specific and domain-independent item generation. An item is generated using domain-independent means if all variable elements in the item can be filled by independent random choices and the answer evaluation is done by plain comparisons with a sample solution. Thus only steps one and four in the item generation process are used in that case. No analysis of input artifacts is performed during the generation process and no constraint choices or additional computations are made. In terms of the taxonomy by Gierl et al., the stem is independent or fixed and answer options are randomly-selected or fixed as well. In contrast to that, an item is generated using domain-specific means if there are dependencies between the variable elements in the item and the rules, master solutions or sample solutions used during the answer evaluation. These dependencies can result from input artifacts, constraint choices or computations based on previous choices. Thus any step in the item generation process can occur during the generation of a domain-specific item, although none of the steps three to five is mandatory. In terms of the taxonomy by Gierl et al., the stem of a domain-specific item is dependent or mixed or the answer options are constrained.

A summary of the different item generation techniques and their usage in steps three to five of the item generation process is given in table 12.1. Not all techniques are applicable in steps three and four of the process, but make at least a contribution to step five. However, no technique is usable exclusively in that step.

Generation Technique	Examples for Usage in Step 3	Examples for Usage in Step 4	Examples for Usage in Step 5	Example Item Types
Ontologies	Select entities	—	Select distractors	Factual knowledge questions, Classification tasks
Other semantic sources	—	Choose random wording for given context	Generate sentences	Mathematical word problems
General purpose software	—	Apply constraint solver to find random instance	Perform complex mathematical or linguistic operations	Mathematical formula problems, Grammatical analysis tasks
Individual algorithms	Analyse source code, models, or alike	Generate meaningful random circuits or trees	Generate or execute program code; analyse generated artifacts	Analytical engineering tasks

Table 12.1.: Summary of different generation techniques and examples for their usage throughout the automated part of the item generation process.

13. Automated Evaluation of Test Item Responses

The evaluation of test item responses is one of the most prominent duties of e-assessment systems and has triggered a lot of research in recent decades. The current chapter will provide an overview on the available techniques with a focus on the integration of domain-specific data sources and tools in the process of grading and feedback generation. Despite the fact that answer evaluation is a very different use case than item generation, many of the techniques discussed in the previous chapter are relevant also in that context.

13.1. Basic Concepts of Automated Evaluation

From the perspective of answer evaluation, divergent assessment items are of particular interest, but also open, convergent ones can require serious effort to generate grades and feedback. Closed items are less interesting, as they have a finite number of possible answers and each of those can be associated with grades and feedback manually. The remaining automated task is then to compare an item response with the list of possible responses, which is a trivial task and requires no knowledge about the domain of the item.

Plain comparisons may also be sufficient in some cases of open, convergent items. If an item asks e. g. for the year in which some historic event took place, then there is usually only one correct answer. Unless the input editor is not very restrictive in accepting other inputs than numbers, then there is nevertheless a virtually unlimited number of possible answers including those that are no year dates at all. For such items, plain comparison is still a sufficient option when it comes to determining the correctness of an answer and if we ignore the usage of "BC" and "AD" in year dates for a moment. But already in these cases it is not sufficient if elaborate feedback should be generated as well. There are several classes a wrong answer can belong to, including a year that is too early, a year that is too late, a year that is in the future, a number that is not suitable to represent a year (such as a decimal number), and anything that contains letters. It is easy for an algorithm to find out which class is the right one for a given answer, but to do so it must be aware of the fact that the answer is meant to be a year date. Hence it must encode domain-specific knowledge about the concept of year dates and also about the current year to sort out years that are in the future.

Following that idea, the generation of grades and feedback is basically a classification task. In particular, there is typically only a limited amount of different possible grades and it must be decided which grade is associated with a given answer. In psychometry, dichotomous items that only know the two possible results "pass" and "fail" are preferred,

but psychometric models also exist for polytomous items [164] that allow for partial credit in face of answers that are partly correct and partly wrong or incomplete. In formative settings such fine-granular marking schemas may be more beneficial [84] and thus the classification task must possibly deal with a large number of classes. The approach mentioned in the previous paragraph above is a rule-based approach to solve the classification problem. It defines a set of rules that refer to characteristics of the answer. In theory, classification can then use the power set of all rules to define possible classes. In practice, the number of classes is typically lower as some rules may be designed in a way that they rule out all other ones. In the example above the rule for answers that are not a number is of that kind. If that rule matches, all other rules do not match, because they expect the answer to be a number. Different to that, rules can also be designed in a way that more than one rule matches for a given answer. In the example above, an answer may be wrong because it lays in the future, but at the same time it is obviously also too late. Nevertheless it is possible to define a mapping that tells which grade is generated for each set of matching rules. That mechanism also works in cases in which rules cannot be provided explicitly. In such cases methods from artificial intelligence such as deep learning with neuronal networks can be used to train a classification model with a set of manually graded answers. The model then learns properties of answers and thus is able to classify also newly arriving answers without using explicit rules.

Some subtle difference can be noticed about the focus of different classification approaches. If only two classes are used, the evaluation checks for correctness and thus classifies into “correct” and “incorrect”. If more classes are used, there can be different meanings. Classes can represent some measure of similarity between an answer and a sample solution. This is a quite neutral meaning, but implies that any part of an answer that is not similar to a sample solution is wrong. More explicit is that meaning if classes are understood as measure of completeness. Different rules can be used to check the presence of some required element and thus an answer can be classified as more or less complete. Again, this implicitly assumes that the presence of some expected part of the answer is a sufficient criterion. Another interpretation of classes can be the meaning of quality. Different properties of an answer can be measured and the classification is based on the difference between a measured value and an ideal one. Concrete examples for each of these different understandings will be discussed in section 13.3 below.

Things can get even more complex if elaborate feedback must be generated that explains the grade. In that case, explicit references to properties of the answer must be included in the feedback, such as quoting a wrong part of the answer or at least naming a specific rule that decided on the grade. Consequently, elaborate feedback needs some kind of parameterized templates or some means of natural language generation similar to the concepts discussed for item generation in the previous chapter. Moreover, additional implicit or explicit rules may be necessary to gather enough information on the answer to create the feedback, which is the basic idea of constraint-based tutoring systems [192]. In the example above, additional rules may check for common confusions of year dates and thus allow to include a proper hint on such a mistake in the feedback. That in turn

means to include even more domain-specific knowledge in the evaluation process up to connecting to an external data source to look up year dates and historic events.

13.2. Preprocessing, Postprocessing and Derived Artifacts

Independent of the way in which grades and feedback are produced, preprocessing of the answer may be necessary depending on the data format used by the e-assessment system. Chapter 11 already mentioned the example input "1/2" that may or may not have the same meaning as "2/4" depending on the context and thus the domain. A specific input format can help to clarify the domain, e. g. by using LaTeX markup to make clear that both input are meant to be fractions. However, the LaTeX format might not be the right format to forward that input to a computer algebra system that is able to compare these inputs with each other or with a sample solution. In turn, the sample solution might look even different, such as like "0.5". While a computer algebra system can easily determine that "0.5" is indeed equal to the fractions "1/2" and "2/4", it may require to transform the input into a different format. Hence, with proper transformation of the data formats, the correctness of an answer can be checked with one single rule in that example. In turn, several rules that check for equality to "0.5", "1/2" and "2/4" separately are needed if less information on the domain is available. Some intermediate processing may be possible as well, e. g. using a tool that is able to shorten all fractions and thus help to solve the case with less rules.

Preprocessing may even be mandatory if general purpose data formats are used. For example, sketch recognition engines can be used to identify shapes in free-hand drawings that are stored in a general purpose format for drawings. Using domain-information during the recognition process can improve the results [9]. The actual grading process then happens on the recognized artifacts and not on the original data format [299]. The strategy to grade derived artifacts instead of the original answer can also be helpful in other cases. For example, answers to items on computer programming can be evaluated by comparing simplified pseudo code or UML models derived from the answers with pseudo code or UML models derived from a sample solution [221, 37]. Additional feedback can be generated by comparing traces from program execution [279]. It may even be necessary to create derived artifacts, if a particular property of the answer cannot be assessed directly, such as the execution semantics of a behavioural model in computer science [280].

Evaluation techniques that work on derived artifacts often can only produce meaningful results, if the input fulfills at least some basic requirements. This observation leads to dependencies between different techniques, where one technique is used to check some basic properties and a second technique is used later on for more sophisticated checks only if the basic checks were passed. The easiest way to cope with that problem is to define a strict sequence of evaluation steps and implement these in a monolithic or single-threaded evaluator (as discussed in section 7.3.3). However, such solutions may be slow if there are many steps that actually have no dependencies. Instead of following a strict sequence, these can be executed in parallel in an multi-threaded evaluator component.

Independent of using evaluation techniques in parallel, postprocessing of the results may be necessary. Different techniques may result in different classifications of the submission and a final grade as well as a final set of feedback messages must thus be compiled. For grades, simple or complex calculations can be used, that do not only use mathematical operations, but may also include conditions and thresholds [96]. Textual feedback can often be merged into a single list of messages. However, in some domains it may be useful to relate messages back to the original answer artifact, e. g. by highlighting wrong parts of the answer. In that case, the individual evaluation technique must produce enough output to precisely locate the correct position for an error annotation.

13.3. Evaluation Techniques

There are several concrete techniques that can be applied to realize the general concept of rule-based tests. As discussed in section 7.3.3 there are several patterns on how to design evaluator components and thus there are many option to combine different techniques. Hence, non of the techniques discussed in the following subsections is meant to be used exclusively and it is in general no drawback if any specific technique is not sufficient to produce the desired feedback completely on its own. Moreover, the same technique can in general be applied in different variants for grading different aspects of an answer.

13.3.1. Evaluation with Domain-Specific General Purpose Software

A nearby solution for the task of evaluating item responses is to formulate the necessary rules as a domain-specific problem and then use an appropriate software to solve it. As already mentioned in section 12.3, many e-assessment systems use computer algebra systems for mathematical items. Most answers to a mathematical item can be understood as a mathematical expression. Computer algebra systems are designed specifically to handle such expressions and to determine the properties of such expressions or the equality of two syntactically different expressions. Thus even in cases in which there are infinite many ways to answer an item with a correct expression computer algebra systems are able to identify these answers and generate appropriate grades and feedback. Moreover, computer algebra systems cannot only produce Boolean results on the correctness of an expression, but can also produce other output that can be used to fill parameters in feedback templates. Consequently, elaborate feedback can even be produced for an infinite number of wrong answers without specific rules by including e. g. an individual counterexample based on the student's input in the feedback.

In some cases, it might be necessary to define a specific way in which answers must be written and it might also be necessary to define complex operations to evaluate the answer. An example for that is the automated evaluation of results from a lab exercise in chemistry [199] where MathLab is used as the underlying general purpose software. Nevertheless the general principle stays the same and elaborate feedback can be given also for the infinite set of inconsistent experiment results, including precise indications for the wrong numbers.

There are also cases in which domain-specific software is designed specifically for the purpose of checking the correctness of an artifact. One of the most prominent cases in that class are coding exercises in computer science, where standard tools for software testing are frequently used for grading submissions [141, 308]. Writing rules for testing certain aspects of an artifact and associating feedback with individual test results is the natural use case of such software tools. Hence they also usually generate elaborate feedback automatically, including counterexamples for failed test cases.

However, the use of domain-specific general purpose software is not always sufficient or helpful, since the software is not designed for educational purposes. For example, in computer science many standard tools exist that perform static checks on source code. These checks do not always produce error messages that are understandable to students and in some cases the techniques used by these tool are even not sufficient to create feedback on the desired level of detail [281].

13.3.2. Evaluation with Generic or Domain-Specific Algorithms

When the use of general purpose software is not sufficient, individual solutions must be created. In fact, it is not possible to draw a hard line between those solutions and the use of general purpose software. Complex operations during grading as mentioned above can be performed with general purpose software, but the more complex these operation get, the more likely is it that they represent a grading-specific algorithm that could be realized in any programming language. In turn, many capabilities of domain-specific software can of course be re-implemented in some programming language. The following paragraphs thus only provide an overview on different option for realizing answer evaluation with general or domain-specific algorithms, while virtually any self-contained evaluator component in any e-assessment system may serve as an additional example.

As it was mentioned earlier above, comparison of a student's answer with a sample solution can be a useful strategy for grading and feedback generation. With some preprocessing, plain textual comparisons with some domain-specific tweaks may be sufficient, e. g. to compare source code based on a simplified representation as pseudo code [221]. If answer artifacts have some graph structure, they can possibly be compared even without preprocessing [262]. However, the more complex an answer artifact or a relevant derived artifact is, the more complex gets the comparison. The necessary comparison strategies and similarity measures can then be implemented in domain-specific algorithms. They can draw from general algorithms e. g. for sequence alignment from bioinformatics [279] or for model matching [290]. Comparisons to a sample solution are especially able to point out omissions in an answer and can thus generate elaborate feedback for an infinite number of wrong answers by listing what is missing. If there is more than one way to answer an item, several sample solutions can be used and the one with a best match with a particular answer is typically used for feedback generation.

A specific issue in comparing textual answers with model solutions can be tolerance in face of typos or grammar issues. In many domains, using the correct spelling is not in the focus of an assessment and hence answers should be accepted as correct, even if they contain spelling errors. In other cases, it may be appropriate to use the same

word in different grammatical forms. There are several different approaches to measure the similarity of words and to identify variants that can be implemented individually or based on existing software libraries [133]. Even more work is required to cope with synonyms, as it may depend on the domain whether two words are considered to have the same meaning [260]. Such techniques are typically used as preprocessing steps before another technique is applied to do the actual answer evaluation. Nevertheless, they can help to reduce the answer space significantly.

In some cases, answer artifacts may be too complex or the variety of possible correct answers too large to create sample solutions and comparison algorithms that compare the answer as a whole. Instead, partial comparisons with larger bits of samples can be used in a rule-based manner. One particular technique that can primarily be used in domains with answers in structured languages with a formal grammar is to use a domain-specific mapping from the answer to a general graph structure as a preprocessing step. It is nearby to do so for answers given in domain-specific languages, since the respective parsers produce a parse tree anyway and can thus be used in a preprocessing step. Once an answer representation as graph exists, graph pattern matching can be used to check for the occurrence of desired or undesired properties (e. g. [291, 268, 225]). Even for natural language that does not follow a formal grammar it is possible do use rule-based approaches, since at least some structures can be detected automatically [162]. The focus of the technique is hence not on similarity with a single sample solution, but on completeness with respect to a list of features that are encoded as individual rules. That typically allows also to accept answers that are an unexpected combination of two different correct solutions. In any case, such approaches require to create more or less extensive rule sets and assign partial credit and individual feedback to each rule. Depending on the complexity of the answer artifacts, it can be hard for item authors to predict all necessary rules. Semi-automated evaluation algorithms can be used in these cases, that present unknown cases to human graders and learn iteratively and adaptively from their decisions [28]. Alternatively, rules can be generated automatically from a corpus of manually graded answer and validated by experts, before they are used in automated grading [312].

Individual algorithms for evaluation are also used in cases in which the actual evaluation task is simple in comparison to the necessary preprocessing or creation of derived artifacts. For example, metrics can be computed for a wide range of artifacts to provide numeric values that represent some characteristics of the artifact [14, 316, 180]. Computing these metrics is the complex steps in that case, while comparing the values to some threshold is easy. Depending on the domain, appropriate thresholds can either be encoded directly in the evaluation algorithms or they must be provided by the item author individually for each item. Notably, the results cannot only be used for explicit classification via thresholds, but also implicitly for the creation of visualisations. Such visualisations can depict the relation between the actual answer and the space of possible answers or the set of correct answers. They can thus be part of an elaborate feedback that is less concerned about an absolute grade but more about the direction in which an answer can be improved [307].

13.3.3. Evaluation with Classification Models

In some domains, artifacts are too complex or are too unstructured to create meaningful results by rule-based checks or systematic comparisons. The most prominent type of artifact that falls into that group are essays. Natural language does not follow a formal grammar and there are very much different ways to express the same meaning even for short texts. In such cases, machine learning techniques can be used successfully [298, 86, 49, 238, 33, 237]. Applications also exist in other areas such as computer programming [111]. In that case, each answer was transferred into a simplified pseudo code representation as a preprocessing step. The general idea in all application areas and independent of some preprocessing is to train a classification model by using a corpus of already graded answers. If that corpus is large enough, the machine learning algorithm can derive implicit rules from it and thus learn to classify unknown texts as well.

While there is much successful research on text classification in general, the use of machine learning techniques for answer evaluation is still limited due to several problems. First, classification based on machine learning works well for a small set of different classes, e. g. a handful of different grades. Precise feedback on errors or omissions in the answer is much harder, as there are typically a lot more classes of different reasons for each particular grade. A classification is thus based primarily on similarity and less on completeness, omissions or particular characteristics. Second, machine learning usually requires a large number of examples for each class and may result in bad models if some classes are heavily overrepresented in the training data set. However, grades are often not equally distributed so that even from large cohorts of students only a smaller set of manually graded answers can be used for training, which makes the process more expensive and time consuming. That also is an obstacle for fine-grained classification, as each class gets smaller the more classes are used. Third, machine learning models cannot by transferred in a simple way from one item to another. Even a minor change in the item task may require to train a completely new model. The same is true for grading with different aspects, where individual models must be trained for each of those. That again makes the used of machine learning models expensive.

To avoid machine learning algorithms to be trapped by irrelevant deviations between different answers, preprocessing may be a very helpful step. Especially for texts in natural language a wide range of techniques can be applied to eliminate typos, different grammatical forms and alike.

13.3.4. Evaluation with Ontologies

In some cases it may not be feasible or even possible to encode all relevant domain-knowledge for answer evaluation within an item. For example, an open item may ask for the name of the current governor or capital city of a given state or country. While it is easy to compare the answer with a sample solution for correctness, it is hard to encode elaborate feedback for all possible wrong, but understandable answers. These include names of former governors/capitals, confusion with neighbour states/countries and confusion with similar but slightly different names. In all of these cases, it may be

worthwhile at least in formative assessments to explain these errors. Ontologies can thus be used to look up additional information for the wrong answer and then draw conclusions on why that wrong answer was chosen. That can also be used to generate partial credit during grading, if e. g. the answer contained a correct family name but wrong given name [270]. Item authors must encode the necessary domain-specific knowledge that makes an error understandable as queries to an appropriate data source. These queries will contain placeholders that are filled with the actual answer during the evaluation process.

13.4. Summary

In conclusion, there is a wide range of different techniques that can be used for the evaluation of item responses. Closed item types are of minor interest in answer evaluation, as the number of possible answers is finite and all possible feedback can be triggered by domain-independent comparison of the response with a sample solution for plain equality. In contrast to that, open item types have an infinite number of possible answers and answer evaluation is thus a classification problem that needs to be solved by domain-specific approaches. Sample solutions can be used here again, but in many different forms: As complete solutions in complex comparison processes, as partial solutions in rule-based checks and as input for training machine learning models. In any case, checking the correctness of an answer to award credits is the simpler task, while the generation of elaborate feedback is harder.

A summary of the different answer evaluation techniques and their key characteristics as well as examples for their usage is given in table 13.1. The focus of each technique mainly determines the cases in which it is useful, while the necessary inputs and helpful preprocessing steps determine the effort needed during item preparation and run-time.

Evaluation Technique	Focus	Input from item author	Possible Preprocessing	Usage Examples
Domain-specific general purpose software	Correctness, Counterexamples	Rule-like expression	Optional; e. g. conversion between data formats	CAS for open math items
Domain-specific testing software	Correctness, Counterexamples	Test case specification	—	Software testing tools for code writing items
Comparison with complete sample solution	Similarity, Omissions	Sample solution(s)	Optional; e. g. spelling correction, replacements of synonyms	Similarity tests for items on conceptual modeling
Comparison with partial sample solutions	Completeness	Rules, partial sample solution(s)	Optional; e. g. spelling correction, replacements of synonyms	Rule-based checks on essays
Comparison of metrics	Quality, Visualisations	Optional; Thresholds	Mandatory to derive metrics	Feedback on quality of text, code or models
Classification models	Similarity	Classification model trained by manually graded answers	Advisable at least for texts; e. g. spelling correction, stemming	Machine learning for natural language grading
Look-up in ontologies	Correctness, Elaborate feedback	References and queries for external datasources	—	Look-up of relations between wrong answer and sample solution

Table 13.1.: Summary of key characteristics and examples for different evaluation techniques.

14. Case Studies

This chapter provides in-depth studies of three different cases of domain-specific item handling within a particular domain. For each domain, a small set of sample items is analyzed that demonstrates the relevance of the four aspects discussed in the previous chapters. Each sample item tackles at least one of those aspects and all items within one case together cover all four aspects. All examples used throughout this chapter refer to sample exercises realized in the e-assessment system JACK that provides substantial implementations for all aspects of domain-specific item handling.

14.1. Case 1: Math

Math education and assessment is a common topic for the use of e-learning systems in general and for e-assessment systems or features in particular. Several systems and components exist that focus solely on math education and assessment (e. g. [188, 187, 241, 110, 235]). Examples of particular architectures and features of such systems have already been discussed throughout part II and part III of the current publication. For the e-assessment system JACK math assessment is one of its main applications and many features have been designed with math applications in mind in the first place [154]. For the following case study, three sample items are examined in detail to demonstrate the relevance of all four aspects discussed in the previous chapters.

14.1.1. Item 1: Equation of a line

The first item in this case study is a quite short item that simply asks students to provide the equation for a line that runs through a given point as shown in fig. 14.1. The item is a typical example of items that are used in analysis in schools. The item has a parameterized stem in which the point is a variable element. It has been implemented in JACK as an showcase for answer evaluation for items that allow for an infinite number of correct solutions. Since the input is already split into two fields that will only contain numbers, no domain-specific input editor or data format is required.

Item generation for that item happens online within JACK right before an item instance is displayed to a user. The only action that needs to be performed by the item generator is to make two constraint choices for the coordinates of the point P. For both coordinates, a range of integer values is defined in the item configuration. No additional calculations are necessary. In particular, there is no possibility to compute a sample solution, since there are infinite many lines that run through a given point.

Consequently, the answer evaluator cannot compare the input to a sample solution. Instead, it has to insert both the random values from the item template and the user

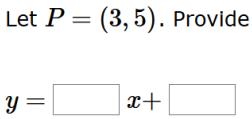


Figure 14.1.: Sample item 1 in case 1. Students are asked to provide an equation for a line through a given point. The answer is evaluated by calculating whether the given point is indeed on the line defined in the answer.

input into an equation of a line and determine whether the equation is fulfilled. Although the resulting expression is easy to resolve for a computer algebra system, it demonstrates that domain-specific operations are necessary due to that fact that the item is open and divergent. Since the answer evaluation only has two possible outcomes (either the line runs through the given point or it does not), no partial credit or additional feedback is given in this item.

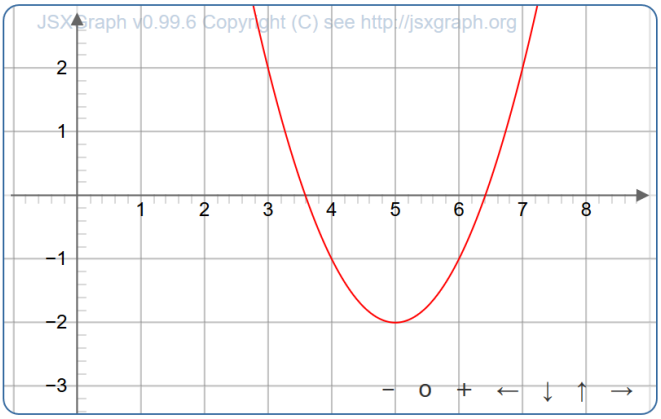
14.1.2. Item 2: Equation of a parabola

The second item is somewhat similar to the first one, because students again have to provide an equation. However, the item stem now not only contains text but a plot of the parabola and the input is entered via a formula editor as shown in fig. 14.2. The parabola shown in the item stem is based on random parameters similar to the point in item 1. The plot is created dynamically and the display is interactive, so that students can pan and zoom the coordinate system to gather all detailed information they would like to see.

The item generation happens again online within JACK, but the process is a bit more complex than for item 1. It starts with two constraint random choices of two integer values for the apex of the parabola. From these, several additional values are derived, including factor b and y-axis intercept c for a parabola in main form. Hence, a complete sample solution is generated in this case. It is not only required for answer evaluation, but also as input for the dynamic generation of the plot included in the item stem.

Answer evaluation also performs several steps. First, the answer can be compared to the sample solution. That is not done as plain string comparison, but through a computer algebra system that checks two equations for equality. Hence, also answers that use an alternative form are recognized. Different feedback and partial credit can be provided if the answer is semantically correct, but not in the requested main form. Similarly, different properties of a wrong answer are checked and elaborate feedback is given for errors like a wrong degree and a wrong y-axis intercept. The former specifically requires to use a computer algebra system that can return the degree of a given equation.

Using a computer algebra system as a prerequisite for the generation of elaborate feedback is not the only remarkable property of the item. The item also stresses the importance of using a domain-specific data format. On the one hand, the answer is represented in proper mathematical notation within the formula editor and on the other hand, it is represented in a tool-specific syntax during answer evaluation. Similarly, the



Have a look at the parabola shown above. Provide the main form of the parabolic equation:

$f(x) =$

Figure 14.2.: Sample item 2 in case 1. Students are asked to provide the equation for a given parabola. Elaborate feedback and partial credit is given based on various properties of the answer.

sample solution is used in two tool-specific variants: One for answer evaluation and one for generating the plot in the item stem. Without integration of domain-specific data formats and tools, that kind of item would not be possible.

14.1.3. Item 3: Estimation theory

In this case study domain-specific tools have so far only been used for answer evaluation and for plotting a graph in the item stem. The third item in the case study now discusses the use of domain-specific tools also for item generation. The item is part of a larger sequence of items on estimation theory and asks students to computer the maximum likelihood estimate for a given sample as shown in fig. 14.3.

The important aspect for item generation here is the fact that the sample needs to be based on a geometric distribution. It is thus not possible to draw the values for the sample with a standard random generator that uses a rectangular distribution. One option would be to simulate the geometric distribution via a domain-specific algorithm that is implemented within the item generator. Another option, that is actually used in that item, is to request the sample from an external domain-specific tool. In case of JACK, the evaluator component can call the statistics software R that allows to request a sample with a single call. Similarly, the same software can be called to calculate the maximum likelihood estimate for a given sample to be used as sample solution.

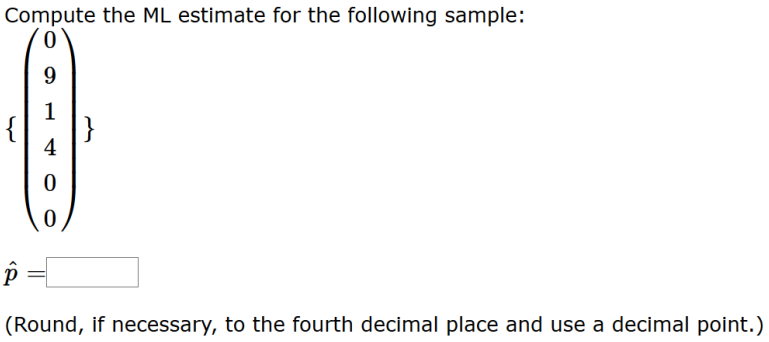


Figure 14.3.: Sample item 3 in case 1. The random sample in the item stem is generated via the domain-specific software package R. Answer evaluation allows for rounding differences by using an error margin.

Since all complex calculations have been performed during item generation, answer evaluation just needs to compare the given answer with the sample solution. Since answers may diverge from the sample solutions due to rounding differences, an error margin is used in this item that allows to accept answers that are slightly different from the sample solution.

14.1.4. Summary of Observations

All four aspects of domain-specific item handling have been discussed in this case study by looking at just three items. Random item generation is used in all three items with increasing complexity: In the first item, simply two random integer values were required. In the second item, additional calculations were required after drawing to random integer values. Finally, the third item required to use domain-specific software to derive a proper random sample.

Since all items are open items, answer evaluation and feedback generation is also an important issue in all of them. However, complexity is decreasing rather than increasing: The first item is open and divergent, so that a domain-specific evaluation is necessary in any case to determine the correctness of the answer. The second item is open, but convergent. Hence, the correctness is mainly a matter of comparison, but additional domain-specific checks are used to create elaborate feedback for various cases. Finally, the third item is also open and divergent, but the answer evaluation simply needs to check whether the answer is equal to the sample solution plus some error margin.

The use of domain-specific data formats and input editors depends much on the situation. While all items include some output in mathematical notation, only one requires a domain-specific input editor.

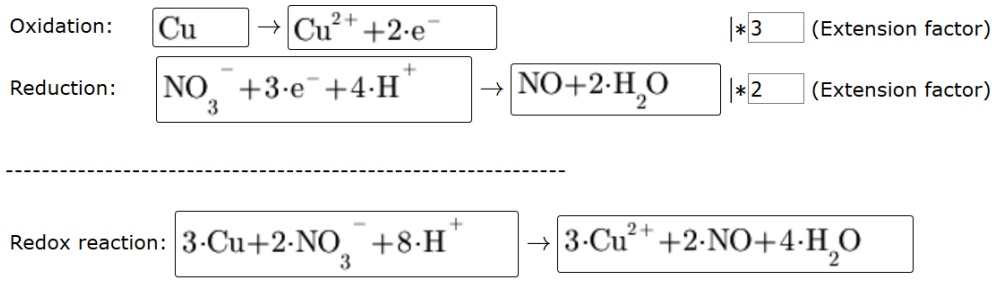


Figure 14.4.: Sample item 1 in case 2. User input is possible via a formula editor that has been designed specifically for chemical formulas. Answers are stored in a domain-specific data format. An evaluator component can evaluate arbitrary expressions that are defined by the item author and that work on the data format.

14.2. Case 2: Chemistry

The second case study is dedicated to the domain of chemistry. As a natural science, it requires a rich variety of competences as well as factual knowledge. Moreover, it deals with several types of domain-specific artifacts and notations. Although the use of e-assessment systems in chemistry education is not as wide-spread as it is in math education, the oldest known system in that context dates back to at least the 1970s [55]. Within the e-assessment system JACK some of its math capabilities have been extended specifically to cover chemical aspects as well, while other features have been added specifically for that domain. Moreover, the assessment of factual knowledge can make use of general capabilities of the system.

14.2.1. Item 1: Reaction Equations

The first item in this case study looks somewhat similar to the ones discussed for the domain of mathematics. Students receive the description of a redox reaction and are asked to provide the chemical formulas that describe the oxidation, the reduction and the whole reaction. Figure 14.4 shows a sample answer to that item. The item has no variable elements, so that item generation is not considered in this case study.

Despite the fact that the input looks similar to a mathematical formula in some way, there are remarkable differences that actually require to use a different data format and hence a modified input editor. Many elements that are known from mathematical notation like subscripts and superscripts occur in chemical formulas as well, but they have different meanings. Hence, the domain-specific language OpenChem has been created for JACK that follows the same principles than OpenMath does for mathematical notations. It thus provides a structured representation schema that can be parsed for processing the input in an evaluator component and an additional representation schema to allow for display and editing in an input editor.

Answer evaluation for the item must be able to analyse several ways in which an answer in any of the input fields can deviate from a sample solution. First, answers can simply be wrong, e. g. by omitting an atom or giving a wrong number. Second, atoms within a molecule can be given in different order, which may violate conventions but is not actually wrong in the first place. Third, the inputs in the last line for the redox reaction may include arbitrary extension factors as long as they are consistent with the other inputs. Hence, an additional check must ensure that not only each input is correct on its own, but also the relation between different parts of the answer is correct.

To achieve that, the evaluator component within JACK has been extended to provide some domain-specific functions. Different to math, they do not represent general relations like “equals”, but relations that have specifically been defined for the purpose of checking the correctness of an answer. For instance, the functions allow to compare the number of atoms between two molecules or check whether two formulas contain exactly the same atoms. Based on these functions, rule-based checks are implemented within the item that assess specific features of the answer and provide feedback for any error or partial credit for any correct aspect.

14.2.2. Item 2: Orbital Schemas

The second item in this case study is concerned with a domain-specific diagram form, namely orbital schemas. These diagrams provide a graphical notation for the electron configuration of an atom, i. e. the number of shells and orbits as well as the position and spin of electrons on these orbits. The electron configuration of an atom follows some fundamental laws and principles and hence it is an important goal in basic studies in chemistry to develop the necessary competences to create and understand orbital schemas. JACK has been extended to support that goal via the realization of a specific editor and evaluation module for items on that type of diagram. Similar to the first item in this case study, item generation is not relevant here. Figure 14.5 shows a screenshot of the editor displaying the electron configuration for Oxygen. Groups of grey boxes with lower case annotations denote orbits, while rows with upper case letters denote shells. The position of an orbit on the vertical axis denotes its energy level. Electrons are depicted as yellow arrows within a grey box with different directions according to their spin.

The range of functions for the editor is relatively small. Users can add and remove shells and orbits and it is possible to place electrons in existing orbits and select their spin. Furthermore, it is possible to remove electrons. For didactic reasons, some simplifications were made with respect to the freedom in diagram layout. This concerns in particular the positioning of the orbits, so that it is not be possible for users to position orbits and shells in a wrong order, nor to assign a wrong energy level to them. Instead, all shells and orbits added to the diagram get automatically assigned to a correct location in terms of energy level. However, it is still possible to add orbits that do not exist in reality (e. g. d-orbits in the K-shell). These orbits also get a conclusive position in the diagram automatically. In order to be uniformly operable on as many browsers and devices as possible, it is possible to operate the editor almost exclusively by "point-and-click", so that neither multiple mouse buttons nor complex mouse movements or finger gestures

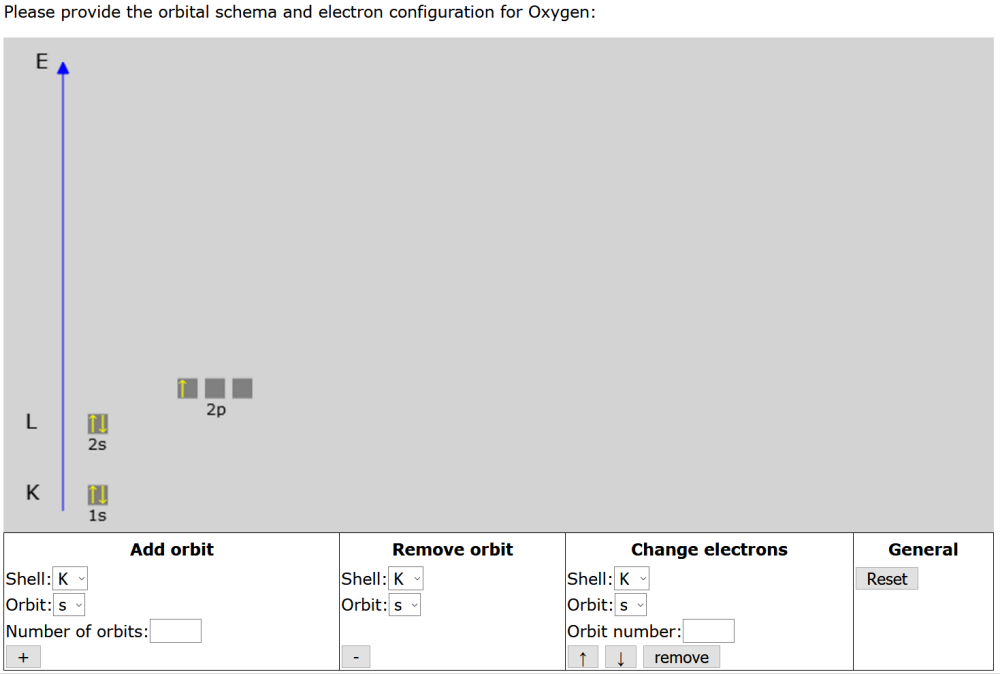


Figure 14.5.: Sample item 2 in case 2. A domain-specific editor allows to manipulate a data structure that has been designed specifically for this item on atom orbital schemas. An evaluator component is able to analyse the data format and provide feedback based on a fixed set of rules.

are necessary. The only exception are two input fields for numbers that require keyboard input. This in turn implies some defaults in that the naming of shells (K, L, M, ...) and orbits (s, p, d, ...) is available in selection fields and thus a completely wrong naming is basically impossible.

Technically, the interaction with the editor manipulates an object structure that represents the atomic orbital scheme. This object structure is then represented graphically in the diagram area on the one hand, and serialized in the form of a JSON string and written into a hidden input field on the web page on the other hand, in order to be sent to the server in this way when a submission is made. The data structure is then analyzed there for automatic feedback generation.

In order to check the correctness of an answer, only the atomic number of the represented element has to be known and some chemical laws have to be considered. It is therefore not necessary for item authors to define individual checking rules for different elements. It should be noted, however, that in general there can be several correct solutions to a task. For this purpose it is only necessary to reverse all spins of all electrons in a given, correct atomic orbital scheme in order to obtain a second, correct diagram. On the other hand, reversing the spin in a subset of the electrons can also lead to incorrect solutions. In certain constellations it is also possible to place single electrons at different positions, all of which are correct. Therefore, different rules must indeed be checked when

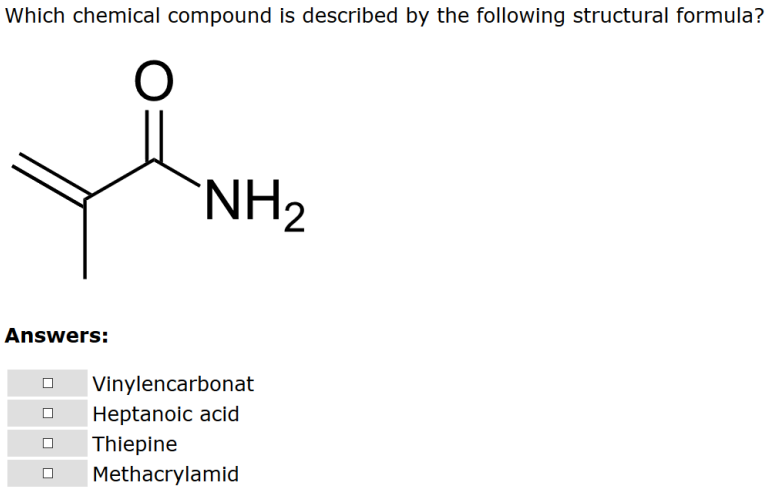


Figure 14.6.: Sample item 3 in case 2. Names of chemical compounds are retrieved from Wikidata, while the figure is retrieved from Wikimedia.

generating feedback and it is not possible to simply compare them with a sample solution. In addition, detailed feedback is generated that names the existing error as accurately as possible, for example by naming the violated chemical law. All necessary rules are implemented directly in the evaluator component. They thus form a domain-specific algorithm for answer evaluation that is defined specifically for that type of item and that operates directly on the domain-specific data format mentioned above.

14.2.3. Item 3: Chemical Compounds

The aspect of item generation has so far been ignored in this case study. Hence, the third item focuses particularly on that aspect. Basic competencies in chemistry include the ability to identify chemical compounds by looking at a graphical representation of their molecule structure. Training sessions thus include items showing molecule structures and asking for the name of the chemical compound like the one shown in fig. 14.6. As it is a tedious task to encode all possible chemical compounds and all their graphical representations manually within an item template or a set of static items, item generation from ontologies is used for that item in JACK. In particular, it makes use of JACK's capability to issue queries to a SPARQL endpoint during the item generation process to retrieve data from an external data source. Wikidata and Wikimedia are used as data sources in this item. The ontology of Wikidata offers entities representing the concepts of chemical compounds, as well as properties describing their chemical structure. However, the graphical representation of the chemical structure is not provided directly by Wikidata, but as a link to an image resource hosted in the Wikimedia Commons project.

The item contains a short question including the graphical representation and offers four answer options, from which one is correct. To achieve that, a SPARQL query asks

for all available chemical compounds and retrieves a list of tuples containing the name and chemical structure. Drawing randomly directly from this list in step three of the item generation process may result in four compounds with very different structures that are too easy to distinguish. Hence, the query retrieves an additional representation of each chemical structure in the canonical SMILES notation. Results are then filtered to only include elements with the same size of the SMILES representation with a predefined length in the range of 6 to 12. This simple heuristic limits the list of query results to ones that are more similar to each other. From this filtered list, the e-assessment system finally randomly draws four elements in step four of the automated generation process. The first of them is selected as the correct answer and hence only for that a reference to the graphical representation is included in the item template.

The heuristic described above may be insufficient to create items of the desired didactical quality in terms of selecting good distractors. For example, the heuristic does not look for molecules including a least some similar atoms. This limitation can be avoided by including more sophisticated, domain-specific filter functions in the item definition to get more fine-grained control of the selected data. Even with very restrictive filters, the number of possible item instances is high, since there exist hundreds of chemical compounds. For each of the possible lengths of the SMILES representation mentioned above, the list of compounds contains 89 to 151 entries. Any combination of four elements out of one of these lists forms a distinct item instance.

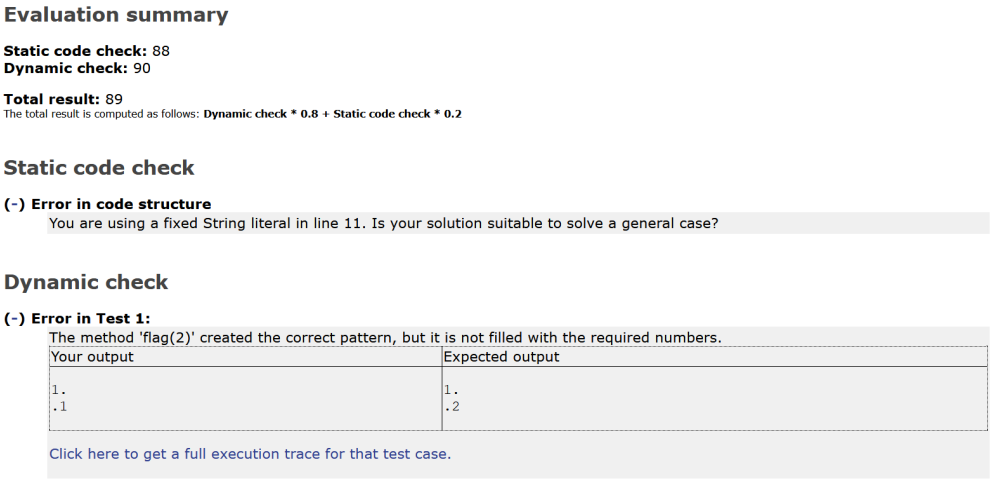
14.2.4. Summary of Observations

The three items discussed in this case study cover a wide range of domain-specific concepts and also make use of all four aspects of domain-specific item handling. The first two items share the same three aspects, but incorporate completely different solutions for input, data format and answer evaluation. Both items make use of a rule-based answer evaluation strategy to cope with the problem of multiple possible correct answers. The first item offers flexibility to item authors via manually defined rules, while the second item allows for a fully automated feedback generation based on minimal input by the item author.

In both cases, domain-specific knowledge is encoded directly within the item or the e-assessment system. Different to that, the third item in this case study makes use of an external data source for domain-specific knowledge. In fact, it would be possible to use that strategy also in the second item and hence create an item in which no explicit domain-specific knowledge is provided by the item author at all.

14.3. Case 3: Computer Programming

The third and last case study deals with programming as part of a computer science curriculum. Despite the fact that programming is also taught in other contexts (e. g. data science or physics), the ability to write meaningful programs and to analyse given program code is primarily considered one of the core competencies of computer scientists. As already mentioned earlier, there are more than 100 e-assessment systems just for the



Evaluation summary

Static code check: 88
Dynamic check: 90

Total result: 89
The total result is computed as follows: **Dynamic check * 0.8 + Static code check * 0.2**

Static code check

(-) Error in code structure
You are using a fixed String literal in line 11. Is your solution suitable to solve a general case?

Dynamic check

(-) Error in Test 1:
The method 'flag(2)' created the correct pattern, but it is not filled with the required numbers.

Your output	Expected output
1. .1	1. .2

Click here to get a full execution trace for that test case.

Figure 14.7.: Sample item 1 in case 3. Results are returned by different evaluator components and a total result is computed by an individual rule. Textual feedback from each component is shown one below the other. Additional feedback is available from the second component via the hyperlink provided in the feedback.

facet of writing program code [141] and the oldest known system for that aspect dates back to 1960 [124]. They are accompanied by a lot more system for other aspects of computer programming, such as code analysis, bug finding and understanding specific algorithms on data structures. The e-assessment system JACK was originally designed also solely for grading items on writing program code and has been expanded later on to cover the other domains discussed above and also other items related to computer programming.

14.3.1. Item 1: Coding Exercise

The item in this case is taken from a programming lecture within a computer science curriculum. The item is one typical example for a multitude of items on code writing that are used during the lecture. The item provides a code skeleton and a task description that asks students to implement some portion of code within the skeleton that causes the program to create a character string that follows a given pattern. The item has no variable elements and thus item generation is not relevant here. Instead, answer evaluation is the most prominent aspect of that item, as JACK is able to produce different kinds of feedback. A sample feedback for an incomplete answer is shown in fig. 14.7.

The e-assessment system JACK provides no integrated source code editor for that item type. Instead, students must use an external editor to open the provided code skeleton and make their additions to it. They can then upload the answer as a source code file. That process can be automated to some extent via a software interface between JACK and the IDE Eclipse, but that IDE is nevertheless an external editor. Consequently,

students have the opportunity to use an appropriate editor that makes sure their answer is at least syntactically correct source code, but they are nevertheless free to submit any kind of file with any content.

Hence, the first step in answer evaluation is to check whether the submitted answer is syntactically correct source code. Any errors encountered in that step are reported as error messages to the student and any subsequent checks are skipped. In fact, that check happens twice, as JACK allows to run evaluator components in parallel, but without dependencies. For the evaluation of this item it uses two component, where one is concerned with static code analysis and one with dynamic testing. Only the component for static checks reports errors during the syntax check, while the other aborts silently.

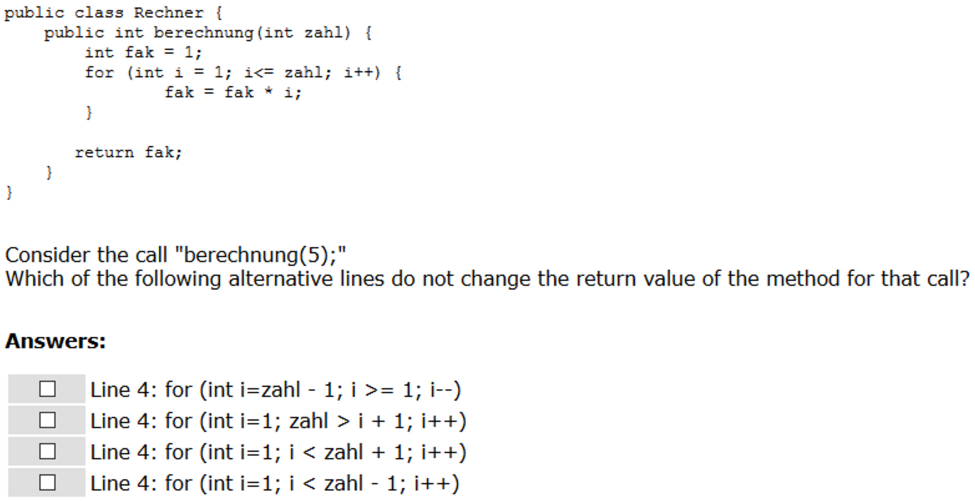
If the answer is syntactically correct, the first evaluator component applies a static rule-based check for code structures. For that purpose it parses the submitted source code into a parse tree and enriches it with additional information to form a slightly more complex graph. A graph query language is then used to execute queries that have been defined by the item author and that are associated with individual feedback. Feedback can be generated either on the presence or absence of some code structure. Elements of the code can be used as parameters within the feedback, e. g. to refer to a specific variable name or a specific line of code.

The second evaluator component compiles the syntactically correct code into executable byte code and performs a series of test cases. Similar to the static checks, each test case is associated with an individual feedback. The actual test results can be included in the feedback, e. g. to show both the expected value and the actual value for a given program call. In addition to that, execution traces of all test cases are created. If a test fails, the respective trace is appended to the feedback message as an additional means of feedback. If the item author also provided a sample solution, difference in the trace of the submission and the trace of the sample solution can be highlighted by an algorithm based on sequence alignment.

14.3.2. Item 2: Code Analysis

The item in this case has also been designed for a programming lecture within a computer science curriculum, but deals with the understanding of source code rather than with its creation. The item is a multiple-response item that presents a piece of object-oriented Java source code in the item stem and four alternative lines of code as answer options (see fig. 14.8). The item stem also includes actual call parameters for the given source code and asks which of the alternative lines of code do not change the program output for that particular call. Hence, the competence students need to demonstrate when answering that item is the ability to analyse program code, deduce program behaviour and compare different behaviours for the similarity of output.

Since the item is a closed item, answer evaluation is of minor interest. In addition, input editors are not relevant, as the item can be answered by ticking checkboxes. The most interesting aspect of the item is its automated generation that involves Java source code as an artifact in a domain-specific language. It is thus also interesting with respect to data formats. The item generation process is split into two stages, where one stage



```
public class Rechner {
    public int berechnung(int zahl) {
        int fak = 1;
        for (int i = 1; i<= zahl; i++) {
                fak = fak * i;
        }

       return fak;
    }
}
```

Consider the call "berechnung(5);"
Which of the following alternative lines do not change the return value of the method for that call?

Answers:

- ☐ Line 4: for (int i=zahl - 1; i >= 1; i--)
- ☐ Line 4: for (int i=1; zahl > i + 1; i++)
- ☐ Line 4: for (int i=1; i < zahl + 1; i++)
- ☐ Line 4: for (int i=1; i < zahl - 1; i++)

Figure 14.8.: Sample item 2 in case 3. Source code is provided in the item configuration, answer options are generated automatically during the generation process.

happens offline within a dedicated item generator outside the e-assessment system, while the other stage happens online right before the item instance is displayed to a student.

The template for the item is straightforward: The item stem consists of just two lines of text with a placeholder for the actual program call including its parameters. In addition, the source code provided by the item author is displayed in the stem. All answer options are placeholders to be filled with a generated alternative line of code. Thus the first automated step in the generation process analyses the provided program code to identify lines that are candidates for meaningful replacements. That is a highly domain-specific task that relies on a rule-based approach to static code analysis encoded in the generation algorithm. For that purpose, the source code provided by the item author is parsed into a parse tree and queried for specific structures. Each of those structures may result in a candidate line for creating a question. From all candidates, the generator randomly picks one or more (belonging to the same method) to be used for generating the actual item. After that decision has be made, the generator knows the corresponding program call and its parameters. Hence, it can assign random actual values to these parameters and execute the code to produce a first output. That is again a domain-specific action. The generator can then create alternatives for the selected lines by making (random) meaningful changes, e. g. switching operands or replacing operators. For each alternative, the modified code is executed again to produce another output. Comparing these outputs with the first one decides whether an answer option is considered right or wrong. Hence, that part of the generation process mimics exactly the same domain-specific behaviour students need to demonstrate when answering the item.

Once all possible alternatives have been created, the offline stage of the generation process is completed. The online stage is much simpler, as it only needs to select four out of the multiple generated alternatives and use them to populate the answer options

in the item template. The number of item instances that can be produced from a single configuration depends on the provided source code, but is typically high. For example, if the source code contains at least one loop-statement (`for` or `while`-loop), several dozen alternative lines can be created for the loop header. Selecting four of them as answer options consequently results in several hundred individual item instances. Notably, not all variants are of equal difficulty, so that results from a careful empirical validation additionally need to be fed into the generation process if item instances with specific psychometric properties are required.

14.3.3. Summary of Observations

Although the two items in this case study are substantially different in the competences they assess, they are strongly connected to each other from a technical point of view by using the same type of data format. In both cases, source code is the central artifact that follows a domain-specific language and can thus be analysed automatically. The fact that the very same mechanisms of pattern matching on the syntax graph and dynamic execution of program code with predefined call parameters is used in both cases stresses that point that the proper integration of domain-specific tools and algorithms is highly beneficial and flexible.

Notably, the both exercises from this case study can be directly connected to each other from the conceptual point of view. The source code submitted by a student as an answer to the first item can be used as input for item generation for the second item. Evaluation results from the first item can be used to make sure that the second item is meaningful, i. e. by selecting an appropriate item template for the second item based on static code analysis results. However, an actual combination of these items would require to realize the item generation process for the second item completely within JACK instead of performing the first stage of the item generation offline in an external generator component.

15. Results

The previous chapters provided an in-depth analysis of the use of domain-specific data sources and tools for input and data storage, automated generation of test item content and automated evaluation of test item responses. The analysis results were also illustrated from a domain-specific perspective by three case studies. Now some conclusions will be drawn to summarize the findings and put them into the context of this publication. Moreover, some connections to the previous parts on processes (part I) and systems (part II) can be drawn.

15.1. Contributions to Integrated E-Assessment

Using domain-specific systems within an e-assessment system is one clearly identifiable aspect of integrated e-assessment. The previous chapters demonstrated that there are different points at which domain-specific data sources and tool can be integrated into e-assessments. Not each of them is relevant for all types of items or all domains. However, all integration points can be used to enrich the capabilities of e-assessment systems. Some cases could be identified in which domain-specific e-assessment is not possible without using domain-specific systems. As a consequence, integrated e-assessment turns out to be a necessary prerequisite for meaningful e-assessment at least in those domains. In turn, integrating domain-specific data sources and tools is an easy way to significantly increase the capabilities of e-assessment systems without adding actual e-assessment features.

This directly relates to the technical perspective of integrated e-assessment that was examined in more detail in part II of this publication. Components like “item generators” and “domain-specific expert systems” that were discussed in that part directly relate to the integration of domain-specific aspects into a general e-assessment system. The results from this part help to identify conceptual interfaces between the generic and domain-specific parts of an assessment item which in turn also help to design proper technical interfaces between system components. The same is also true for the design of data storage within an e-assessment system that may respect the different structures of domain-specific data formats. Finally, the discussion on domain-specific evaluation of item responses strongly advocates for modular design patterns for graders in which domain-specific tools are combined as needed in order to assess different details of a student response.

However, there is not only a technical contribution. For each of the four aspects (input, formats, generation, evaluation) summary tables provided an overview on different classes of techniques or occurrences of the respective aspects. They can be used to classify assessment items or assessment systems and compare them to each other based on their domain-specific capabilities or demands. The results on several integration points can

thus be used as a starting point for a systematic analysis of domain-specific e-assessment systems and their capabilities. For each domain, a first step is to check whether it actually uses specific input and data storage formats, general purpose software, or information systems. A second step is to check whether e-assessment items make use of them as well. A third steps is to identify domain-specific tools that can be used as content or question generators or response processors. Any gaps between availability and actual usage identified in the second and third step can then be used as direct triggers for further development towards more evolved e-assessment items and systems that reflect the full capabilities of a particular domain.

While this procedure for advancing e-assessment systems for a specific domain is a process on its own, there are also additional connections to the general assessment process discussed in part I of this publication. The use of domain-specific data sources, formats and tools is clearly coupled to specific alpha states from the Assessment Kernel. In particular, state "Designed" of alpha "Test Items" refers to resources belonging to a test item and may thus include data in domain-specific formats. Moreover, the state "Designed" may be the one that is reached after domain-specific automated generation of items has been done. Similarly, state "Generated" of alpha "Grades & Feedback" may be reached after domain-specific methods for automated evaluation have been applied. Finally, also state "Defined" of alpha "System" is coupled to domain-specific aspects, since the connection to external, domain-specific systems may be an important requirement when defining which e-assessment system is used for a particular assessment. As a consequence it can be stated that the use of domain-specific data sources, formats and tools has a direct connection to several phases of the assessment process.

15.2. Contributions beyond the Scope of this Publication

There are much more applications to domain-specific data sources, formats and tools than their use in e-assessment systems. The structured discussion in the previous chapters can thus also be used as a starting point to analyse the same aspects for other types of e-learning systems. In particular, virtual laboratories and similar training systems may be of interest here, as they are also supposed to make use of highly domain-specific data. The four summary tables provided above can thus serve as blueprints for similar classifications of domain-specific and domain-independent features and demands in related systems and environments.

At the same time, the structured discussion may even be of interest for system developers in areas other than education, as long as template-based content generation or domain-specific input and data formats are concerned. For example, the generation of quests and riddles within a virtual world of an adventure game can use the same techniques. The insights from any deeper analysis on using the same techniques across systems and application areas can then be used to improve the robustness of formats and tools and to ensure their universal use, which in turn also improves their usability for integrated e-assessment. In conjunction with the connections to system design from part II this can also be used to define and realize middleware components that are specifically

designed to fill the gap between e-assessment systems or other technology-enhanced learning systems on the one hand and domain-specific general purpose systems on the other hand.

The results from this part of the publication may also be used to compare domains with each other from a data perspective and identify similarities and differences. Based on that, data formats and generation mechanisms can be generalized and unified independent of their actual use in the context of a specific e-assessment system. This in turn helps to identify clear interfaces and components for the system design and helps to improve the overall software quality. More specifically, it helps to identify which concepts and constructs are the same in different domains and which can complement each others. For example, different natural sciences make use of basic mathematical concepts, but extend or apply them in a specific way, as it has been shown in one of the case studies in the previous chapter.

Part IV.

Data-focused E-Assessment

16. Data Produced by E-Assessments

It is quite obvious that e-assessment systems do not only receive data in terms of tasks and configurations provided by item authors as well as answers submitted by students, but also produce data in terms of grades and feedback. The mechanisms for the latter as well as the system components involved in the related processes have been discussed in the previous parts of the publication. Moreover, the conceptual view on grades and feedback as main data elements has already been explored earlier in part I. However, there is no need to restrict the creation of grades and feedback to a direct computation for each single item with a subsequent simple aggregation to compute a score for the whole test. Psychological research has come up with a lot of procedures to gather insight into a person's capabilities and competences by using well-designed tests, in which the feedback generation for an individual item is less important. Hence, one of the topics in the current part of the publication will be the integration of such methods into the context of e-assessments and how to use them for sophisticated approaches such as adaptive testing.

The discussion on the educational assessment process also revealed that there might be more data of interest to teachers or learners. Consequently, the use of suitable metrics to compute characteristics of test items or tests had been included in the checkpoints for alpha "Test Items" (see table 3.1) and alpha "Test" (see table 3.2). While that refers to the use of data within the assessment process, i. e. to improve the assessment quality, there might also be interest in using data from a single assessment in a wider context and thus outside the actual assessment process. Examples include the improvement of lectures or learning materials [70], the selection of courses to take in the next term, or the prediction of success. Moreover, teachers might be interested in simply exploring detailed results from several assessments, viewing the performance of different students in comparison or query data for aggregated results of specific groups of items within an assessment.

A fundamental difference in both topics can be made by the scope of data usage: One scope is to gather and analyse data to learn more about a single person or a group of persons, while another scope is to gather and analyse data to learn more about a single item or a group of items. However, the underlying data is always produced by individual persons and many methods produce results that reveal insights about persons as well as about items. Hence, each of the following chapters discusses groups of methods that share the same approach or that are connected to each other in some way.

Even more than the previous parts of the publication, the current part is concerned with the integration of e-assessment into a larger context. Psychological research has already been mentioned above and provides a lot of results that can be integrated with e-assessment on the application level, i. e. for adaptive assessments. Similarly, the research area of "learning analytics" and "educational data mining" has produced a lot of results

on how to use data from various e-learning systems and thus naturally also covers the use of e-assessment data [7]. There are also approaches for "assessment analytics", that can be organized in a conceptual framework with input, processes, output and feedback [81, 211]. The most important idea in that framework is the fact that assessment happens in some context that determines what needs to be analysed, why, when and where it needs to be analysed, and who needs the analysis results. Consequently, assessment analytics can be seen as some kind of interface that integrates the actual assessment process that happens within an e-assessment system with the educational context in which the assessment is embedded. However, integration is not only an aspect on the conceptual level of contexts, but also a plain technical issue. Entirely different concepts for storing data (such as relational databases, XML data, and RDF stores) might be used across different systems and hence data export or transformation is necessary before data from different sources can be used [178].

Additional aspects that cannot be neglected in any of the different context facets are trust and legal issues of data privacy. When students take part in an assessment, they reveal explicitly and implicitly private data and thus must trust the assessment system and the assessors to handle that data properly. In turn, assessments can be used to certify knowledge or competences and thus must make sure that accurate and detailed data on individual persons is available, which usually involves to record and store private data. It is not possible within the current part of the publication to dig into legal issues, since they may depend on national laws and local regulations. However, a brief summary of the main aspects of using private data is included before the details of data usage are discussed. Finally, this involves also ethical aspects. Again, the following sections will only report on some main aspects that are discussed in more detail and with a broader scope than just e-assessment in other publications (e. g. [259, 147]).

16.1. Literature Study

To get a clearer picture of the various contexts in which data from e-assessment systems can be used, a simple, systematic search in the Scopus literature database has been performed. The search terms *(e-assessment OR "computer aided assessment" OR "technology enhanced assessment") AND data* have been used for that purpose. Additional terms like *automatic assessment* have been tested as well, but produced too much irrelevant results from areas other than educational assessment.

Search terms were applied to titles, abstracts and keywords of the papers in the database. In addition, the search was limited to articles, conference papers, book chapters, and reviews. A first run of the search was performed in March 2021, followed by a second run in October 2021 to include more up-to-date results. Since data for 2021 cannot be complete before the end of the year, only papers published in 2020 or earlier are considered.

Figure 16.1 provides an overview on the classification process that has been applied to the search results. Classification of all papers was done based on the content of title, abstract and keywords. Classifications were recorded by assigning at most two labels

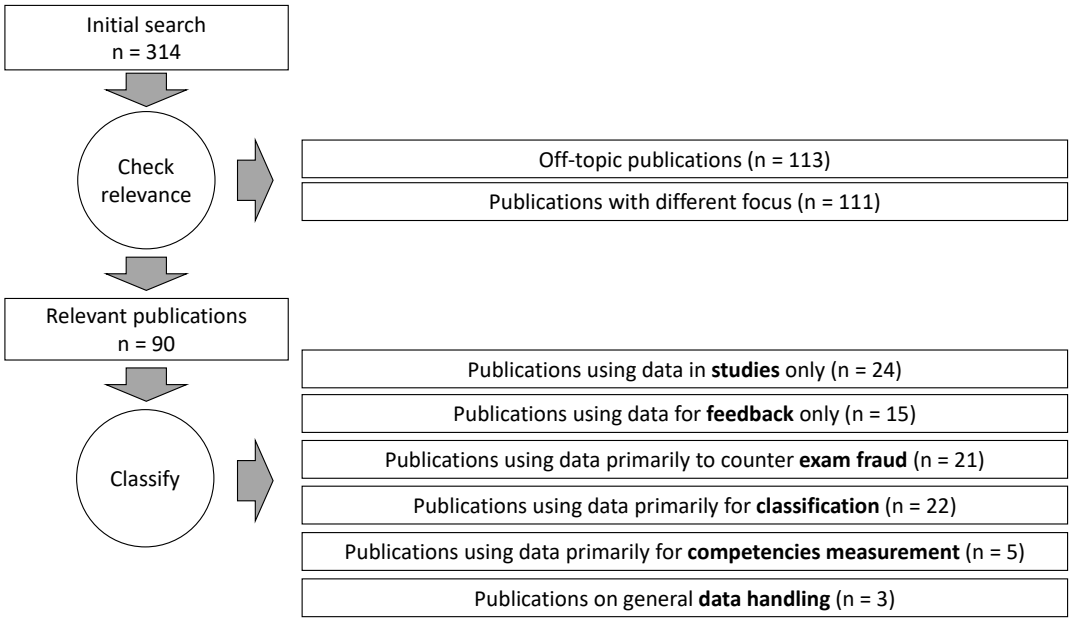


Figure 16.1.: Overview on the classification process applied in this literature review.

to each paper, denoting the primary topics of each paper. Papers that did not cover the topic of data usage in or from technology-enhanced assessment or even were not concerned with educational assessment at all were labeled with label *"off"*. These papers are excluded from any further investigations. Five additional labels were used for papers that covered technology-enhanced assessment, but did not focus on data usage:

- Label *"study on e-assessment"* was used for papers that discussed studies on e-assessment where data was not taken from the e-assessment systems, but solely from other sources such as surveys or interviews. A different label *"data use in study"* was used for papers that discussed studies on e-assessment where data was indeed taken from the e-assessment systems and that where thus considered relevant for the review.
- Label *"system design"* was used for papers that only presented and discussed system design but not specifically data handling. A different label *"data handling"* was used for papers in which system design was discussed with an emphasis on data handling, which was considered relevant for the review.
- Label *"review"* was used for papers that presented reviews of other publications but did not originally report on the use of data within some system.
- Label *"theory on e-assessment"* was used for papers that discussed general and abstract theories on e-assessment or process models for e-assessment, but did not discuss the actual use of data.

Label	Papers	Label	Papers
off	113	data use in study	24
study on e-assessment	46	feedback	18
system design	44	plagiarism	8
theory on e-assessment	9	authentication	8
review	6	quality	8
domain-specific item handling	6	privacy	7
		improvement	6
		prediction	6
		dishonesty	5
		competency measurement	5
		adaptivity	5
		classification	5
		data handling	3

Table 16.1.: Number of papers per label for excluded papers (left) and included papers (right). Note that papers from the right-hand part could be labels with more than one label and thus the sum of all labels is larger than the total amount of papers.

- Label *"domain-specific item handling"* was used for papers that are concerned with domain-specific e-assessment in domains that involve the term "data", such as "data structures" in the context of computer science.

For the remaining papers, at most two labels were assigned the describe best the primary kind of data or means of data handling contained in that paper. There was no pre-defined list of possible labels before starting the review, but new labels were defined as necessary. To decide which label(s) can be applied to a paper, title and abstract were read first. In most cases, these contained sufficient information to select one or two appropriate labels. In case of doubt, the full paper was read to make a decision.

The search returned a total amount of 314 publications. From those, 224 publications were excluded with the labels listed in the left-hand part of table 16.1. The remaining 90 papers were assigned with labels as listed in the right-hand part of the same table.

16.1.1. Bibliometric Data

Publication years (see figure 16.2) seem to reveal an increasing interest in the topics that are considered relevant for the review. The oldest publication is from 2005 and thus fairly new (at least in comparison to the oldest excluded search result which is from 1948) and there were not more than two relevant publications per year until 2010. Different to that, there were at least 6 relevant publications per year since 2014 and more than 50% of all relevant publications have been published within the last four years.

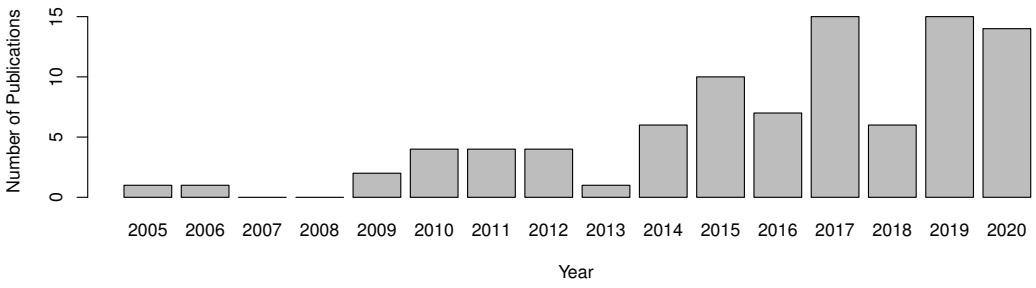


Figure 16.2.: Number of papers per year that have been included in the analysis.

The most frequent publication venue among the relevant papers is the IEEE Global Engineering Education Conference (EDUCON) with five papers. It is followed by two journals (British Journal of Educational Technology and International Journal of Emerging Technologies in Learning) and two proceedings series (Lecture Notes in Computer Science and Lecture Notes on Data Engineering and Communications Technologies), each with four publications.

16.1.2. Findings on Contexts and Forms of Data Usage

The 90 relevant publications that were included in the study can be divided into several groups. These will be discussed in the following paragraphs in decreasing order of their size.

The largest group contains 24 publications that report on some research study on e-assessment, where at least a part of the data used in the study comes directly from an e-assessment system. In comparison to the large body of research in the field of technology-enhanced assessment this seems to be a small number. It may thus not be representative, but only a random sample that contains all search terms by chance. Nevertheless, it can be concluded that using data from an e-assessment system for research purposes is at least a very relevant use case, if not indeed the most frequent one. Common to most of the studies is that data is usually extracted once. The focus of research is usually not an individual person, so that anonymous or aggregated data can be used. However, persons must remain identifiable if data from e-assessment systems should be combined with data from other sources such as interviews or questionnaires.

A major group of 21 papers is concerned with measures to detect or prevent exam fraud. A significant share of the publications in that category origin from the recent "TeSLA"-project[1]. Both aspects are related closely to each other, but can be distinguished by the way they use data: One aspect is the *detection* of plagiarism and other forms of academic dishonesty. A total amount of 11 publications tackle that topic and discuss approaches on how to detect dishonesty from e-assessment data. The topic is not specific

[1] `https://cordis.europa.eu/project/id/688520/`

to technology-enhanced assessment but also relevant for paper-based exams. However, technology-enhanced assessments make it easier to analyse solutions (cf. e. g. [209]) and to collect additional information like keystroke characteristics [26] to reveal dishonesty. At the same time, unproctored e-assessments may make it easier for students to commit exam fraud (cf. e. g. [12]). With the respect to data usage, anonymous data can be sufficient to check for indicators for exam fraud and personal data must only revealed in conjunction with actually suspicious cases. Some mechanisms combine data from e-assessment systems with other data (e. g. previous submissions of coursework or resources from the internet; [30]). Moreover, checks can be run once (e. g. after an exam) and need no constant access to data.

The other aspect is authentication and privacy, which can be used to *prevent* exam fraud. It is discussed by a total amount of 14 publications. The important trade-off here is how much personal data must be revealed to ensure a proper authentication and how much data can be kept anonymous [207, 196]. Different to the previous aspect, data is usually used continuously (e. g. to make sure that the person who logged in for an exam or enrolled for a course is indeed the person that works on the exam/course) and in conjunction with external data sources (e. g. for single sign-on mechanisms).

An almost equally large group of 22 publications is concerned with data classification approaches that employ mathematical or statistical models. There is a wide range of application areas: Adaptive e-assessments (e. g. [239, 34, 99]), prediction of exam results or completion time (e. g. [45, 297, 92]), or quality measurement and improvement (e. g. [263, 22, 69]). Pure classification can also be used as a means of data aggregation for feedback generation [200, 240]. Besides adaptive e-assessment, these aspects are mentioned in the papers several times in conjunction with the term "learning analytics". Different to the research studies on e-assessment mentioned above, the focus of the publications is not on a detailed measurement that is performed once in the context of academic research, but on continuous or frequent use of data for the respective purposes.

Although the mathematical or statistical methods might be similar for different purposes, the kind of data is not. Adaptivity clearly requires continuous use of individual data, since such systems adapt their contents based on individual responses while learners are working on an assessment or assignment. In contrast to that, prediction is usually performed frequently and can also involve data from other sources such as learning management systems. Data used for quality measurement and improvement is usually aggregated and anonymous, while data that should help students to improve their way of learning in personalized systems clearly needs to be related to that person [243].

The next group contains 15 publications that are concerned solely with giving feedback on individual items (while there are two papers that are not only concerned with feedback, but also with classification and another paper that tackles feedback and plagiarism). Since 15 papers is a rather low number given the fact that most e-assessment systems are primarily designed to give feedback, these papers can hardly be considered representative for the way in which e-assessment data is used for feedback generation. Nevertheless, these papers already show that feedback generation requires continuous usage of data. If feedback is solely directed towards the learners, feedback mechanisms can use anonymous data. Feedback for teachers that reports about a larger group of learners can use

aggregated data, but feedback in exams obviously is related directly to individual, identifiable persons.

A special aspect of feedback generation is competency measurement, for which 5 publications could be discovered that are explicitly related to that topic. Similar to the low number of papers on general feedback generation, it is possible that much research on that topic is published without direct relation to e-assessment and has thus not been discovered by the search terms used for this simple survey. An interesting aspect with respect to data handling is the fact, that competency measurement not only uses data from e-assessment systems, but also produces data (i. e. measured competency levels) that may be stored as additional data directly associated with individual persons in some kind of learner model [42, 85].

Finally, 3 publications discuss general topics of data handling within e-assessment systems independent of a particular use case. The discussions cover meta-data management [242], conversion between data formats [179] and approaches to data visualization [191].

16.1.3. Discussion of findings

The results prove that there are various views on e-assessment data that all get remarkable attention in current research. Notably, no time constraint was used during the literature search and most papers have been published fairly recently. The oldest publication included in the results is from 2005 and there are only four publications at all that have been published before 2010. Given the fact that e-assessment systems are known for much longer, the focus on data use appears to be a relatively new topic.

The results of the survey are similar to the results of a broader survey on artificial intelligence applications in higher education [313]: In that survey, profiling and prediction was identified as a major use case (58 out of 146 studies), followed by assessment and evaluation (36 studies), intelligent tutoring (29 studies), and adaptivity and personalisation (27 studies). Hence only the aspect of authentication and privacy was not covered in that survey, which is not surprising as these topics are usually not associated with the use of artificial intelligence.

Besides a classification into topics, that literature survey can also help to identify characteristics of data usage along different dimensions. One dimension that was already mentioned above is the frequency of data use. Data can be extracted from an e-assessment system one time for single use, i. e. in the context of a research study. It can also be extracted or used frequently. This is the case for example when solutions are checked for plagiarism at the end of an exam or when data is extracted at the end of a course for quality assurance. Finally, data can also be used continuously, e. g. for adaptivity, competency measurement, or during authentication.

Another dimension is the granularity and richness of data. For many studies or for quality assurance it is sufficient to use anonymous or aggregated data that does not reveal too much individual details. Also grading and feedback generation can often be performed without revealing personal data of the answer's author. Anonymous data is particularly beneficial with respect to data privacy. Aggregated data is more compact to handle than detailed data and thus e. g. easier to visualize. Other scenarios like competency

Type of data	Cases with one-time use	Cases with frequent use	Cases with continuous use
Anonymous or aggregated data	research studies	quality assurance	feedback
Individual, identifiable data	—	plagiarism check	adaptivity, competency measurement
Data merged with external sources	research studies	plagiarism check, prediction	authentication

Table 16.2.: Overview on examples for typical scenarios using e-assessment data along the two dimensions of data usage.

measurement or adaptivity nevertheless require individual, identifiable data since they concern individual students. Using anonymous or aggregated data is not possible in that case, although that means to involve more sophisticated algorithms to handle large amounts of data and to ensure data privacy. In very specific cases, particularly in conjunction with extensive research studies, but also for prediction, plagiarism checks or some ways of authentication it may be necessary to combine e-assessment data with data from other sources. That can be achieved by contributing data to a general data repository. The resulting data is very rich and detailed, but also very sensitive with respect to data privacy.

An overview on the two dimensions of data usage and some scenarios is given in table 16.2.

16.2. Legal Issues, Trust and Privacy

In the first place, e-assessment systems gather sensitive personal data. Even a small set of answers given by a student can reveal much about his or her competences. Judgements based on that data may have remarkable consequences for the future life of that student. Consequently, handling e-assessment data has not only technical and didactical implications, but also implications with respect to laws on data privacy (such as the European General Data Privacy Regulations (GDPR)) and trust into e-assessment systems. A usual procedure in that context that is required by many laws is to ask data subjects (i. e. students) for informed consent in using their data. Data controllers (i. e. teachers or technical units providing e-assessment facilities) must provide the data subjects with complete and understandable information on the purpose of data processing, data types, storing period, data controllers and processors, and the data subject's rights. Blueprints exist for a structured presentation of these information [197]. However, more legal requirements have to be taken into account, especially if it is not voluntary for students

to provide some personal data. In particular, e-assessment systems are neither allowed to collect any data that is possible, nor are teachers allowed to use the collected data for any purpose they like. Moreover, collecting, processing and storing data implies some obligations for the data controllers and processors.

16.2.1. Technical and Organizational Considerations

A first and probably quite obvious aspect in that context is data security. E-assessment systems and its users must make sure that data cannot be manipulated within the system or leaked to unauthorised users outside the system. Both technical and organisational measures can be implemented for that purpose and they are in general not different than in other systems that deal with sensitive personal data. Suitable measures thus include the use of TLS to secure the communication between components within the e-assessment system's distributed architecture, the use of a PKI to avoid transmission of passwords through the network, or the strict use of authentication between all components within the architecture, as suggested for the architecture created within the TeSLA project [143]. In addition, block chain technology can be used to save exam results and resulting certificates from manipulations, even if they leave the system [112]. Notably, system architecture can also have an impact on data security as we already discussed in section 7.3.1 in part II: The *unrestricted plug-in* pattern bears that risk that critical data may leave the system and thus data security cannot be guaranteed when that pattern is used.

A second and more delicate aspect is the balance between authentication and privacy. Teachers must make critical decisions based on e-assessment data and thus naturally want to make sure as much as possible that students cannot cheat the system and produce data that does not reflect their actual competences. The problem of ensuring identity authentication can be considered to be the main challenge to be addressed for remote e-assessment [143]. Complex authentication mechanisms, collection of additional data besides the actual answers to assessment items and online proctoring can be used for that purpose to prevent exam fraud. However, that in turn implies to collect and process a very large amount of personal data and students may not agree with the appropriateness of that data collection. In fact, students may expect that an e-assessment system is as neutral as possible and thus will grade answers anonymously. Indeed, there is no technical reason to store any other personal data than the actual answers within an e-assessment system for the purpose of assessment and competency measurement. The problem can partially be solved via technical means like pseudonymous identifications, where standards like OpenID, SAML or Shibboleth are used [159, 143]. However, that only allows to keep personal data local during the authentication process, but does not remove the need for further data collection by online proctoring or other means of authorship verification.

A third aspect is that of data usage beyond actual assessment. As reported above, there is a large amount of research studies that make at least partial use of data from e-assessment systems. While it is nearby to do so in research on e-assessment, there is no automatism that this is also legal. In particular, a signed consent form for all students is only usable if it indeed informed about their actual use of the data. Hence,

any use of data that is not mentioned at the time a student gives consent is not allowed. Consequently, teachers, researchers, and operators of e-assessment systems must decide and publish what they want to do with data before they collect it. Otherwise, they can only use anonymized and aggregated data, that may be sufficient for many but not all types of research studies.

16.2.2. Ethical Considerations

Besides technical and organizational considerations, processing and storing personal data also implies ethical aspects. Two major ethical aspects have been explored within the TeSLA project [147]: A first aspect is the freedom of giving consent. A consent in processing personal data is only free, if a true alternative is offered. If students for example do not trust in an e-assessment system that uses face recognition for authentication, it is not a true alternative to offer voice recognition for those student who give no consent in taking pictures of them. On the other hand, offering them the alternative to take a classical face-to-face exam without an e-assessment system bears the risk of unequal treatment in two completely different types of exams. In addition, there are possibilities to allow students to give partial consent, so that their data can e. g. be used in scientific studies (where an individual cannot be identified from aggregated data), but not for personalized learning [259]. The idea here is to create a balance that allows to use anonymized data for the benefit of the majority, but also allows individuals to protect themselves from potential harm.

A second aspect is transparency and feedback. The way personal data is processed within an e-assessment system must be understandable to students and also to teachers. Students must be able to understand the algorithms that process their personal data within the e-assessment system to be able to challenge decisions made by the system. In particular, it must be transparent to them which decisions are actually made by a system based on which data, and which decisions are made by teachers using the system. Similarly, it must be clear to teachers how far data processing results produced by the e-assessment system (e. g. a plagiarism warning) can be understood as evidence or just as a hint. It also must be clear for which purpose the feedback produced by the system is actually intended to prevent teachers from over-interpreting data or using it in context for which it is not suitable.

Another important ethical aspect is that of the temporal nature of student identity and performance [259]: Each assessment or other recording of student data can only provide a snapshot for a specific time and context. Consequently, longitudinal data is necessary to come to more stable conclusions that do not overestimate the value of a single observation. At the same time, storing of data should be limited, so that older and possibly outdated data can only have an limited impact.

17. Competency Measurement

In most cases, the main goal of an assessment is to make the (intermediate) result of a learning process measurable in some way. The easiest way to do so is to count the number of assessment items that were solved correctly within an assessment and use that number directly as measurement result. Since that barely reflects the fact that items can be of different complexity or difficulty, a common approach is to assign some kind of weights to each item. In any case, the underlying assumption is that an assessment item allows to capture a direct, representative picture of the actual abilities of the test taker. This is what classical test theory (cf. e. g. [204]) is about. Besides the assumption that a person's psychometric construct can be directly measured it also acknowledges that errors can occur in a test and that thus only an observed score can be measured. Consequently, classical test theory analyses the relations between the actual score, the observed score, and the error of a person.

A more modern view assumes that learning results in a gain of competences, where a single competence is defined in [305] as "the cognitive abilities and skills possessed by or able to be learned by individuals that enable them to solve particular problems, as well as the motivational, volitional and social readiness and capacity to use the solutions successfully and responsibly in variable situations." A subsequent assumption in [146] is, that "Competencies cannot be reflected by or assessed in terms of a single, isolated performance. Rather, the range of situations in which a specific competence takes effect always spans a certain spectrum of performance. Narrow assessments cannot meet the requirements of competency models. [...] Competence must be assessed by an array of tasks and tests that do more than simply tap factual knowledge".

Consequently, sophisticated item preparation and data analysis is necessary if e-assessment systems should be used for proper competence measurement. In turn, data produced within e-assessment systems can easily be used to analyse assessment items more closely and thus come to detailed conclusions about the actual competences that are related to a particular item. For example, it must be assured that the items within one particular assessment that is supposed to measure a single competence or a set of closely related competences requires predominantly certain levels of these competences. This requirement closely resembles the criterion of Internal Consistency known from classical test theory, which can be calculated by Cronbach's Alpha Coefficient [60]. Alpha will be negative whenever there is greater within-subject variability than between-subject variability. The common rule of thumb for Internal Consistency is "excellent" for $\alpha \geq 0.9$, "good" for $\alpha \geq 0.8$ and "acceptable" for $\alpha \geq 0.7$. Another example is the fact that competence measurement establishes a relation between the competence level of a person and a difficulty of an item. That implies that it is no longer sufficient to assign random

weights to items, but to calculate mathematical models that ensure a proper alignment of item difficulties.

The following sections first summarize basic knowledge about competency measurement using item response theory (IRT). It then shows how that can be used for adaptive testing within an e-assessment system.

17.1. Item Response Theory

In contrast to the classical test theory, item response theory assumes a psychometric construct to be latent and only observable through responses on items that are solved by a particular probability related to the person's ability. In general, item response theory assumes that the difference between a person's ability and the difficulty of an item is a predictor for the probability of an individual's response [223].

A basic assumption in IRT is that an item is dichotomous, which means that an answer is either right or wrong. The probability for a correct answer (an outcome of "1" in the following) can then be expressed by a function of the item difficulty (β) and the person's ability (θ):

$$P(X = 1|\theta, \beta) = f(\theta, \beta) \tag{17.1}$$

The difficulty of an item is hence assumed to be the value a person's ability must have to have a 50% chance to solve the item. Moreover, the chance decreases monotonically with decreasing ability and increases monotonically with increasing ability. A typical approach to characterize the required function is to use so-called "logistic model" where the calculation is based on the transformed logit function:

$$p(z) = \frac{e^z}{1 + e^z} \tag{17.2}$$

where z has to be specified more exactly.

17.1.1. One-dimensional Logistic Models

As mentioned above, the probability of solving an item is determined by the difference of a person's ability θ and the difficulty of the item (β). This leads to a formula representing the probability of a response of 1 as a function of item and person parameter.

$$p(x_j = 1|\theta, \beta_j) = \frac{e^{(\theta-\beta_j)}}{1 + e^{(\theta-\beta_j)}} \tag{17.3}$$

$p(x_j = 1|\theta, \beta_j)$ is the probability of a response of 1 on the jth item with regard to a person parameter θ and the item difficulty β_j. The model is called one-dimensional, because difficulty and ability are each considered to be a single value.

Obviously, neither the difficulty of an item nor the ability of a person are known a priori. Instead, both values can only be computed if the answers of a group of persons to a set of items is known. This is why item response theory is particularly interesting in conjunction with e-assessment systems, as all data is available digitally. Data can thus

easily be exported in a data format suitable for popular statistical software, which offers the necessary functionality to perform all required calculations (e. g. the "Psychometric Models and Methods" for R[1]).

The first and simplest form of the test models related to the item response theory is a model where only one parameter is to be estimated. This model only differs in the difficulty of the items and is called the 1PL model or Rasch model. Additionally, it is assumed that the latent variable is one dimensional. The first step in the calculation process is then to fit the model to the data, i. e. compute β for all items based on the given answers.

If the interest is only in fitting a logit function to empirical data, it is necessary to change the discrimination value α of the *item characteristic curve* (ICC) to better fit the data. Nevertheless, for the 1PL model the discrimination value has to be the same for all ICCs of the investigated items. Including the discrimination value into Equation 17.3 leads to the general formula describing the 1PL model.

$$p(x_j = 1|\theta, \alpha, \beta_j) = \frac{e^{\alpha(\theta-\beta_j)}}{1 + e^{\alpha(\theta-\beta_j)}} \tag{17.4}$$

The 1PL model has several assumptions concerning the items. They have to be locally stochastically independent, homogeneous and have to fulfill specific objectivity. Besides these assumptions and the restriction on one parameter, the resulting person and item parameters are versatile. Thus, the complete set of parameter values can be shifted without changing the probability of solving an item. Because of that, comparing the person's parameters among different tests is not applicable.

Provided that the 1PL model is applicable, some very convenient simplifications can be made. For example, the sum of the scores of all individual items is a sufficient statistics, which means that the (estimated) person parameter depends only on the total number of correct answers of this person. It does not matter, which items the person had responded to correctly.

In addition to the 1PL model, there are other models that, as a commonality, do not require the presumptions of the 1PL model. In the 2PL model – also called the Birnbaum model after its author (cf. [35]) – a second parameter δ is introduced for each item. It describes the gradient of an item and thus its discriminability. Discriminability describes how well an item is able to discriminate between persons with different ability levels. If discriminability is low, a person with low ability has only a slightly lower chance to answer the item correctly than a person with higher ability. If discriminability is high, persons with insufficient ability clearly have a low chance to answer correctly, while persons with sufficient ability clearly have a high chance. The Birnbaum model is described by the following formula, which introduces the parameter δ_j for α in Equation 17.4:

$$P(u_{ij} = 1|\theta_i, \beta_j, \delta_j) = \frac{e^{\delta_j(\theta_i-\beta_j)}}{1 + e^{\delta_j(\theta_i-\beta_j)}} \tag{17.5}$$

[1] `https://cran.r-project.org/web/views/Psychometrics.html`

Different values for item discriminability are reflected in the steepness of the slope of the item characteristic curves. Consequently, the item characteristic curves can intersect each other and there is no longer specific objectivity. This would mean that the difficulty order of the regarded items depends on the person parameter.

One of the assumptions so far is that the chance to answer an item correctly is close to zero for persons with low ability. While this may be true for open items, it is usually not true for closed items that can be answered by guessing. To correct for that factor, a third parameter can be introduced, resulting in a 3PL model. That parameter basically changes the y-intercept of a curve and thus defines a lower bound for answering the item correctly that is substantially larger than zero.

17.1.2. Item Information Curves and Test Information Function

In the previous subsection the item characteristic curve (ICC) was used to depict the chance of a person with a given ability to correctly answer an item with a given difficulty. The first derivative of an ICC is called *item information curve* (IIC). It shows how much new information can be learned about a person with a given ability by knowing their answer about that item. Item information curves have their peak at the difficulty value and thus at the point where the item has the highest discrimination. A person with that ability has a 50% chance to answer that item correctly and thus both outcomes are equally likely. In turn, a person with much lower or higher ability has a much lower or higher chance to answer that item correctly. Thus it is more likely that the expected answer is indeed the actual one and thus the gain of information is lower to both sides of the difficulty value. This is the core mechanism used for adaptive testing that will be discussed in section 17.2 below.

Item information curves can be used by assessment authors to optimize their tests. Two items with the same IIC are redundant for the purpose of competence measurement. Instead, one would try to have items in the test that have difficulty values that are equally distributed over a wider range of abilities. The sum of all IICs can be used to visualize the coverage as the *test information function*. One would expect that function to have its peak around the expected average ability and to be moderately decreasing to both sides.

17.1.3. Beyond One-dimensional Models

In the case that there is more than one psychometric construct to be measured within a single item, multidimensional models have to be applied [234]. Another approach can be to split complex answers into segments that are locally stochastically independent and hence handle one assessment item implicitly as a set of individual items, where each of those measures only one psychometric construct. An example on that is discussed in section 20.2 below.

17.2. Adaptive Testing

A classical exam presents a predefined set of items to the participants. Depending on the context, the order of the items may be random for each participant (if an e-assessment system is used that allows to shuffle items) or participants can at least work on the items in random order. E-assessment systems may also draw items randomly from a larger item pool so that participants do not only get the same items in different order, but also different but similar items. However, these mechanisms still do not take into account how a student has answered the items. Given the fact that each item provides most information about a person's ability if its difficulty is equal to the person's ability, that implies that many participants may answer items that actually do not help much in measuring their competences [151, 90]. Moreover, in an assessment that covers a large range of ability or difficulty levels, it can be expected that the easiest items will be answered correctly by most participants, while the hardest ones will be answered wrong by most participants. This may have a negative impact of the performance for many participants, as they are either bored by the easy items or demotivated by the hard ones.

A solution to that problem are adaptive tests, in which each participant receives an individual sequence of items based on the performance during the test. In adaptive testing, based on previous answers, the participants selectively see only those items from a larger item pool that match their ability level and that are necessary to measure a desired competence as accurately and validly as possible. The central goal of adaptive testing is thus to limit the testing to those items that are informative for the desired ability measurement [151]. In adaptive testing, students are thus confronted as little as possible with items that are significantly above or significantly below their individual learning level and thus provide little diagnostic information. Using adaptive testing tailored to individual learners in assessments or exercises also offers opportunities to respond to a relatively heterogeneous groups of students in a course by offering individual learning support [189]. Hence, adaptive testing can be used both in summative scenarios for a final exam and in formative scenarios for exercises or informative self-assessments. Due to the increased test efficiency of adaptive tests, such self-assessments can be short but still reliable, which may increase the motivation of learners, as they receive quick and individual feedback [47].

In order to cope with the extensive statistical evaluation when selecting questions, adaptive tests are often carried out digitally by using an e-assessment system. They are commonly called "Computerized Adaptive Tests" (CAT) in that case. The statistical test evaluation approach of the item response theory (IRT) as discussed in section 17.1 is primarily used here and in fact the term "adaptive testing" is almost directly associated with that method [151]. With this approach, one is able to determine item difficulty and personal ability parameters of an item, which are the basis for the adaptive selection of items in the test. However, items that can be evaluated automatically are required, which is why closed item formats are particularly suitable. This can lead to restrictions in the measurement of certain psychometric constructs or competence facets, e. g. with regard to performance for the independent creation of solutions. In that case, open items are more suitable, but also more challenging from the perspective of psychometry.

In addition, adaptive testing requires a sufficiently large number of scaled items. In particular, this type of testing requires extensive item pools that cover the full range of expected abilities with a large number of item to measure an ability in a way that is appropriate for the model. Moreover, an extensive calibration sample is required to estimate the corresponding item parameters sufficiently accurately and reliably [90].

These psychometric requirements are often difficult to meet in the case of complex tasks and specific target groups. This variant of adaptive testing, also known as "tailored testing", is based on an elaborate calculation to estimate the ability parameter and to select the next task that provides maximum information. Above all, this approach also requires the fulfilment of the above-mentioned prerequisites (extensive task pool and extensive calibration sample) in order to be able to perform corresponding calculations of the parameters. A variant that provides less demanding prerequisites for the development of adaptive test items or tasks are "branched testings", in which the performance-related item selection is made according to a previously defined branching structure [152]. This variant of adaptive testing places significantly lower demands on the statistical evaluation and calculation of item parameters during the test procedure. The basis of such test variants is a decision structure for the adaptive assignment of individually appropriate items, which is elaborated in a subject-related didactic or learning theoretical manner. It is nevertheless based on the individual performance during the assessment, as each item is seen as a decision point that may lead to a different, predefined branch based on the correctness of the answer.

18. Learning Analytics and Outcome Prediction

Besides measuring competences with calibrated assessments according to psychometric theories, teachers or students may also be interested to get other insights into a learning process. In particular, they might be interested in measuring learning progress without performing a psychometric valid assessment every day or week. E-assessment systems offer the opportunity to gather more low-level data such as the performance in individual exercise tasks and to export that data to make connections with data from other sources. Teachers and students can use that data either to analyse the actual learning progress within a specific group of students or to come to more general conclusions about the appropriateness of some learning activities in a given course. That in turn can e. g. be used by students to predict whether their own learning effort is sufficient to reach their goals or by teachers to see whether their class is on track. For a simple approach to that task, dashboard or analysis tools are sufficient that allow to explore data from various sources, e. g. based on an ontology framework [136] or as a detailed dashboard with both numerical and textual information [245, 98].

Statistical methods for performing more sophisticated analyses and predictions instead of ad-hoc inspection of data are summarized as "Learning Analytics" [142, 254], sometimes also mentioned in conjunction with the term "Educational Data Mining" [7]. These methods are concerned with a statistical analysis of data from learning processes as well as with its visualisation. Learning analytics are not limited to data from e-assessment systems, but can include virtually any data that can be recorded throughout a learning process. However, assessment scores are used by about a half of all current approaches to learning analytics according to a recent literature survey [250]. Another study specifically on predictions using Support Vector Machines recorded six (out of ten) cases in which a final mark was predicted from previous marks [20].

The following sections thus focus on learning analytics methods that can be used to analyse or visualise assessment data. Other methods like interaction data and social network analysis or text mining from student contributions to chats or discussion boards are not considered here. These can be used e. g. for gathering insights into structure and behaviour of learning groups [194] or for predicting whether students will complete their assignments on time [75]. Although the latter is related to assessments, it is out of scope here as it predicts events within an assessment but solely uses data from the context. Instead, we focus on cases in which data from assessments is used. Hence, statistical methods are summarized that allow to cluster students based on their assessment performance and to predict learning outcome at the end of a course by early assessment results. Two classes of methods that are used particularly often are clustering

and regression [250, 7]. The purpose of clustering is to predict the class assignment of objects (or persons) based on some observed properties. The number of classes or clusters is typically substantially smaller than the number of elements to be placed in these classes. In contrast to clustering, regression can also predict a continuous size or floating point number and can thus assign an individual value to each element. Nevertheless, it can also be used for classification. In both case, a mathematical model must be created from training data that contains both the input values and the desired output values for a sufficiently large sample. The required model can be created with classical statistical methods or with artificial intelligence, i. e. neural networks or other forms of learning systems. Frameworks exist that implement several classification methods and thus allow teachers to explore data and choose the most appropriate method [175]. Once an appropriate model is in place, predictions can be made for a new data set.

The techniques described in the following sections are the ones that are commonly used in several studies (see e. g. [233]). Specifically, a recent survey on artificial intelligence applications in higher education lists several studies that use artificial neural network (see section 18.3), decision trees and random forest (see section 18.5), support vector machines (see section 18.4), naive Bayes (see section 18.2), and regression analysis (see section 18.1) to predict admission decisions, student academic performance, drop-out undergraduate students, and student engagement [313]. Notably, that survey names logistic regression to be conventional and counts all other techniques as machine learning algorithms. Another recent survey lists twelve different categories of data mining techniques in higher education, where the four techniques classification, clustering, statistics, and regression are the techniques used in 72% of the reviewed studies [7].

18.1. Regression Analysis

Regression analysis is a classical statistical analysis method with many application areas. The two main variants are multiple linear regression and logistic regression. The latter is also called binary logit model. Both methods attempt to determine the relationship between one dependent variable and several independent variables. With multiple linear regression, the assumption is made that the relationship is linear. Furthermore, the dependent variable must be interval-scaled and the independent variables must be interval-scaled or dichotomous variables. In a graphical visualisation of the method, each element is considered a vector in a vector space (where the number of dimensions is equal to the number of variables). Within that space a straight line is sought that describes the relationship between dependent and independent variables in such a way that the distances between the straight line and the independent variables are minimized. To do this, the squared deviations of the straight line from the independent variables are minimized - this procedure is also known as the "method of least squares" or "Ordinary Least Squares" (OLS). The resulting regression model is described using regression coefficients:

$$y = \beta_0 + \beta_1 * x_i + \beta_2 * x_2 + ... + \beta_k * x_k + \epsilon$$

In this equation, y is the dependent variable, x_k are the independent variables , β_k are the regression coefficients for each x_k, and ϵ is an error term. The equation thus directly expresses a linear correlation such that an increase of x_k directly causes a change of y in the magnitude of β_k, provided all other independent variables did not change.

If the dependent variable is categorical data, that is, the variable can only take a limited number of instances, linear regression cannot be used. In this case, logistic regression is used. In contrast to multiple linear regression, the dependent variable in logistic regression has only two instances, while the independent variables can be both interval-scaled and dichotomous variables. Similar to multiple linear regression, a function curve is sought to describe the relationship between dependent and independent variables. However, in logistic regression, a logistic function is used instead of a straight line. Here, the logistic sigmoid function is used for the probability estimation so that a high or low probability is output for one of the two instances of the dependent variable. This is based on the method of "maximum likelihood estimation" (MLE), in which the regression parameters are determined in such a way that the desired high or low probabilities are calculated. In contrast to multiple linear regression, which predicts a concrete value of the dependent variable, logistic regression thus predicts the probability of the occurrence of one of the two possible values for the dependent variable. However, there are two other variants of logistic regression analysis. Namely, ordinal logistic regression and multinominal logistic regression, which can be used for ordinally scaled dependent variables and for nominal dependent variables with more than two values.

The results of a regression analysis are usually easy to interpret and visualize. In addition, it is offered as standard functionality in many statistics programs or even simple office software, which makes it easier to perform regression analyses. However, regression analysis does not always provide high-quality prediction results. For example, individual outliers can have a strong influence on results and independent variables with a high collinearity also decrease stability of results [289]. Furthermore, complex, non-linear relationships are difficult to capture using regression analysis. Nevertheless, the regression analysis is sufficiently understandable, so that it is often used for prediction and visualization in the field of Learning Analytics. A detailed case study on that is included in section 20.1.

18.2. Naive Bayes

Naive Bayes or the Bayes classifier is an algorithm for classification problems and is based on Bayes' probability theorem. The algorithm is mostly used for object classification by using probabilities to predict the assignment of a new object to one of several known classes. For this purpose, relevant properties of the objects are seen as random variables. The distribution of the properties within a training sample with known classes for each object can then be observed. Hence, the Bayes' Theorem can be used to determine the probability $P(A|B)$ that an object belongs to class A if it has property B by observing the general probabilities $P(A)$ (the share of objects in class A) and $P(B)$ (the share of

objects having property B) as well as the probability $P(B|A)$ for finding property B on an object of class A within the training sample:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Notably, the assumption that all attributes or properties of the objects are stochastically independent is rarely true in reality. Nevertheless, the assumption is used in order to reduce the complexity of the problem. Because of this naive assumption, the Bayes classifier was named Naive Bayes. Another assumption is that numerical properties follow a normal distribution, which is also a quite strict requirement that cannot always be fulfilled in reality.

Nevertheless, Naive Bayes is an algorithm that is often used in practice, as it offers numerous advantages. It can be calculated quickly and can easily deal with missing attributes or properties of objects. In addition, the algorithm can be implemented with little effort. If the assumption about the stochastic independence of the object attributes is valid, Naive Bayes can perform even better than logistic regression, but requires less training data at the same time. In addition, good results can be achieved with Naive Bayes if input data sets are available as categorical variables that have a limited number of instances. Finally, Naive Bayes learns incrementally, which means that a more accurate prediction can be made with each newly classified data set. Additionally, this algorithm can be used to predict the probability of multiple classes of target variables.

The main disadvantage of the approach is that it is based on an assumption of stochastic independence of the object properties. Thus, if there are in fact strong dependency between attributes of the classified objects that are neglected, the resulting classification is less meaningful. Nevertheless, meaningful results can be obtained by using Naive Bayes, as long as object attributes are not too strongly correlated. For high-dimensional problems with many object properties, the method also becomes inefficient. Moreover, the completeness of the training data with respect to all possible values of categorical variables must be observed. Assigning a value to a categorical variable that did not occur in the training data set leads to the 'Zero Frequency' problem because the probability is assigned a value of zero and hence no prediction can be made.

The Bayes classifier is often used to identify the behavioral patterns of learners. For this purpose, user profiles can be included, which contain information about the number of exercises performed, online times, topics covered of learning content, exam results, and chat and forum participation [246].

Baysian classifiers can be somewhat hard to understand and interpret for students due to the concept of conditional probabilities. Bayesian networks can be used to represented the classifier as a network structure that makes the model understandable for the learners. For example, Bayesian networks can be used with log data from an e-assessment system to predict whether a question is answered correctly or incorrectly in the system [311].

18.3. Artificial Neural Networks

Artificial neural networks are a method of artificial intelligence inspired by the structure of the human brain. They are based on simple arithmetic units, which are supposed to correspond to the neurons of the human brain and which are organized in several layers. Each neuron receives a vector of input data and calculates an output value that is part of the input vector for the next layer. Thus, artificial neural networks are essentially based on matrix calculation and are thus suitable in principle for all problems where the input data can be organized in the form of matrices or vectors and a vector can be interpreted as an output, which is particularly true for classification problems. Artificial neural networks can be used to solve a wide range of application problems in statistics, engineering and economics. In contrast to Naive Bayes, which is used exclusively for classification purposes, artificial neural networks are also used for regression analyses.

Neurons form the main component of artificial neural networks and are also called units or nodes. A distinction is made between input, hidden and output neurons. Input neurons can be used to record input data in the form of signals, stimuli or patterns. Hidden neurons are positioned between input and output neurons to represent the internal information pattern. The number of hidden layers can be varied, although in practice networks with only one hidden layer are often used. Output neurons transmit the results obtained to the outside world. The neurons are connected by edges from one layer to the next, thus creating a layer architecture. The structure of artificial neural networks can be organized in different ways, which is why a fundamental distinction is made between feedforward networks and recurrent networks (also called feedback networks). In the first approach, information from input to output neurons can be processed exclusively in a forward direction. In contrast, in recurrent networks the information can flow in a backward direction; i. e., the nodes already visited can be traversed again. In any case, each edge is assigned an individual weight, which determines how strongly the information conveyed via it influences the respective target neuron. Since the actual behavior of the neurons is trivial and identical for all, these individual weights are the decisive factor by which the artificial neural network builds the classification or regression model.

During the training of neural networks, the weighting of connections between neurons is therefore changed by means of learning rules or algorithms. The training can take place as supervised or unmonitored learning. In supervised learning, a certain output is specified and the weighting of neurons is adjusted based on the comparison between target and actual state. In contrast, in non-monitored learning, no output is specified and the weighting is optimized based on different inputs. A training phase is followed by a test phase in which the learning success of the artificial neural network is determined.

The fields of application of neural networks are manifold, but they are used particularly often in cases where a strictly mathematical modelling (such as via conditional probabilities in the case of Naive Bayes) is not possible, either because there is not enough knowledge about the structure of the problem or solution, or because the input data are usually fuzzy or incomplete. The most common applications of artificial neural networks therefore include in particular image, pattern, script and speech recognition as

well as early warning systems or prediction models that operate on particularly large and high-dimensional data sets. Here artificial neural networks can achieve good results in complex or non-linear problems. A major advantage of the method is the high generalization capability of the networks, which means that after a successful training phase even unknown inputs can usually be processed well. Furthermore, neural networks have a certain error tolerance, so that despite internally occurring errors the functionality is maintained. Furthermore, artificial neural networks are capable of learning and adapting due to their training capabilities.

However, training also shows the disadvantage of the approach. In order to train artificial neural networks sufficiently and thus achieve reliable results, a large amount of training data sets is required. In particular, these data sets must cover the width of the problem or the solution space as evenly as possible, since otherwise an artificial neural network runs the risk of systematically excluding underrepresented areas. With unsuitable training data, artificial neural networks can thus develop bias in favor of overrepresented properties. Furthermore, there is the risk of over-adaptation by storing exactly the training patterns and not abstracting them anymore, which makes the method lose its ability to achieve useful results even on unknown inputs. As another and more general drawback, the decisions made by artificial neural networks are difficult to reproduce and explain, since the networks behave predominantly like a black box. In addition, the training of artificial neural networks often requires powerful computers and multiple, time-consuming training runs. In contrast to direct implementations of a strict mathematical model, artificial neural networks are also significantly slower and more ineffective in performing their calculations.

In the context of e-assessment, artificial neural networks are suitable methods for recognising patterns in the learning behaviour of students. For example, artificial neural networks can be used to predict students' success in a course by using log data and exercise results from a learning management system. In particular, grades, number and length of online times and their percentage over a certain period of time can be included in the analysis [44] and missing values for single students cause no substantial problems [91]. On the one hand, this procedure allows to determine with a high statistical probability which user profiles are successful and which are not. On the other hand, the procedure is so complex that it is difficult for teachers to understand, making it also difficult for them to provide personalised feedback to learners [311]. As mentioned above, there is a risk of a systematical bias within an artificial neural network if training data is not evenly distributed. As grades and other performance related indicators tend to follow a Gaussian distribution, extremely good and extremely bad students are usually underrepresented in the data. However, these students may be the ones who would benefit at most from clear hints about their predicted performance. Moreover, courses that require a characteristic learning behaviour (e. g. due to the way in which learning materials and exercises are published throughout the course) but have a low number of participants often do not produce enough data at all for using artificial neural networks.

18.4. Support Vector Machines

Support Vector Machines (SVM) are mathematical algorithms for classification or regression analysis. Basically, the method is used to divide a large number of objects into two classes, for which the objects are represented as vectors in a vector space. The method then attempts to place a separation plane in this vector space that runs exactly between the objects of the two classes and maximizes the distance to the objects closest to the plane. To do this, SVM is first trained using training objects for which the class assignment is already known, to determine the separation plane of the two classes. The vectors closest to the parting plane are called support vectors and give the method its name. Vectors further away from the plane do not have to be considered for the calculation, so that only a small part of a large amount of training data has to be considered. In reality, however, objects are rarely linearly separable, which is why a separation plane cannot always be calculated. To be able to use the method nevertheless, the non-linearly separable data is artificially extrapolated to a higher-dimensional vector space, in which a suitable separation plane can be inserted.

Typical examples for the use of SVM in practice are image, font and face recognition, spam filtering, handwriting recognition and applications in the field of bioinformatics. An important advantage of SVM is its high generalization capability, thanks to which the support vector machines can be used to solve practical problems and overfitting of the model can be avoided. Support vector machines allow a quick classification based on parameters based on a few support vectors instead of entire training data sets. In doing so, the functionality of SVM can be represented geometrically, which is why the results are transparent compared to the black box results of neural networks.

Nonetheless, SVMs have significant disadvantages. For example, it is not possible to extend existing results. Therefore, a new training is required as soon as new input data sets are available. Furthermore, the SVM method is comparatively slow and requires a lot of memory in the learning process. The larger the number of classes, the more memory is required. In addition, knowledge of the necessary dimensional size is assumed when converting non-linearly separable data in higher-dimensional hyperspace.

Similar to artificial neural networks, support vector machines are complex methods for recognizing certain learning patterns based on user profiles. For example, SVMs can be used to classify learners into two groups (pass/fail). Good predictive results could be obtained based on important factors such as attendance, learning time, number of errors and preliminary grades [172, 311]. It is also possible to use SVM as a classifier for several classes. In this case, for example, for predicting grades, where each grade should be considered as a class, so training data is needed for each class [20].

18.5. Decision Trees

Decision trees are a method by which a large number of objects is divided into classes with regard to a predefined target value and according to their properties. If the dependent target value is available as a nominally scaled variable, the method is referred to as a

classification tree. If, on the other hand, the target variable is a quantitative variable, it is referred to as a regression tree [167]. In both cases a decision tree has a hierarchical structure consisting of a single root, several inner nodes, leaves (also called outer nodes) and edges. The root and all inner nodes each represent a decision, which usually refers to exactly one property of the objects to be classified. The basic idea of decision trees is therefore to make a tree run, starting at the root and making a decision at each inner node, until a leaf is reached that specifies the classification of the object.

Different algorithms can be used to generate a decision tree. One example is the Classification and Regression Trees (CART) algorithm, which was already presented in 1984 by Breiman et al. as a univariate, binary decision tree [38]. A CART algorithm is used to generate a binary decision tree, which divides the training set into two disjoint subsets based on a pair of values. Since the classification is based on a few instances, there is usually a high variance of the decision tree.

The CART algorithm has both advantages and disadvantages. The decisive advantage of the method results from the ability to distinguish relevant from non-relevant properties of the objects to be classified. For this purpose, a preprocessing is performed to calculate the relevance of each property. This makes it possible to create decision trees for objects with many properties that come to a decision with little depth. CART is also easy to understand and interpret, which is why the results are transparent, unlike the black box results of neural networks. The algorithm is stable against exceptional cases and can also handle incomplete input data. However, CART also has disadvantages, for example by generating unstable decision trees, since even a small change in training data leads to enormous changes in the decision tree. Furthermore, splits are only performed one-dimensionally. This means that only the comparison between a variable and a threshold value is made in inner nodes for decision making.

Problems with the high variance of a decision tree have a negative effect on the ability to generalize. Therefore, in practice, an extension of the traditional decision tree model such as Random Forests is often used, which is based on a multitude of decision trees. For this purpose, the training data is divided into arbitrarily chosen subsets and the decision trees are trained individually. To classify the objects, they are first classified individually from all decision trees. Finally, the class assignment is made following the majority of classifications or the average of the probabilities of a class assignment. Random Forests thus provides a more stable prediction result and delivers more meaningful results than traditional models such as decision trees or SVM. However, this requires more training data to be able to create a sufficient amount of decision trees.

In practice, the number of students in a course may actually be too small to make meaningful use of the procedure [176, 311]. Nevertheless, if the amount of training data is sufficiently large, the method provides good results, for example, in identifying students at risk in time and evaluating their performance. Information such as grades, attendance, lab work, and submissions can be collected to predict exam performance at the end of the semester [208].

19. Item and Answer Analysis

The previous chapters discussed aspects of data processing that are directly related to learning and assessment. Besides these aspects, data collected by e-assessment systems can also be used for other purposes that are not directly related to the performance of an individual student. Closely related to the quality of a submission is the question whether that submission was indeed produced by the student. While electronic systems make cheating and dishonesty easier, they also make it easier to apply algorithms to e-assessment data to detect plagiarism.

Competency measurement and learning analytics also implied to measure some properties of assessment items to produce valid mathematical models. Item analysis can continue on these ideas and use e-assessment data to gain further insight into characteristics of assessment items and thus help teachers to produce better learning materials. While most of these analyses are not limited to e-assessments, electronic systems help to collect required data automatically and speed up calculations. Notably, the fast availability of these analyses can even change the character of an assessment: If teachers can for example easily see that a significant amount of students struggles with a particular assessment item, that does not necessarily tell much about the individual students, but a lot about that item and the associated lessons. Hence, additional data analysis can shift the focus of an assessment from assessing individual persons to an assessment of educational quality that allows to correct issues in the educational design early.

19.1. Plagiarism and Authorship

Plagiarism in terms of including some part of an other person's work into the own submission is probably one of the most common forms of dishonesty in assessments. Different types of plagiarism can be identified [10]: In "literal" plagiarism, students include an exact copy, a near copy (some sentences are deleted, inserted, joined or split) or a restructured copy from another document into their own work. In "intelligent" plagiarism, they use more sophisticated text manipulations like paraphrasing or summarizing texts, using automatic or manual translations from documents in another language, or adopt ideas. Notably, it depends on the actual assessment task whether all these cases are indeed considered plagiarism. In particular, if students are e. g. asked to present a well-known theory with their own words, they are actually asked to adopt an idea and hence that is not considered plagiarism. Moreover, not all types of plagiarism are applicable to any type of artifact. While plagiarism of source code is a well-known problem in programming assignments [57], source code cannot be plagiarized by summarizing or paraphrasing. Also restructuring needs to be done very carefully and is thus somewhat harder to perform on source code than on natural language texts.

While copying, restructuring or translating larger portions of text or any other kind of artifact manually in a hand-written exam at least requires some time, copying digital texts or other pieces of work as well as using automated translation tools is quite easy and thus even more appealing to students in electronic assessments. At the same time, it is also easier to check digital submissions to assessment items for plagiarism than hand-written documents. Hence, checks for plagiarism can easily be integrated into the grading process of e-assessment systems.

If plagiarism detection is performed on a set of documents by comparing each to the others, it is also known as "external" or "extrinsic" plagiarism detection, while finding suspicious sections within a document based on some properties or characteristics is also known as "intrinsic" plagiarism detection [218, 10]. The latter is also known as "forensic analysis" [171]. The following subsections discuss both types of analyses in more detail.

Most approaches to tackle dishonesty by plagiarism detection are only applicable to open, divergent assessment items that allow students a large degree of freedom in designing their answer. They are thus well suited for (short) essay items and can partially be applicable to items in some formal languages like programming or modelling tasks.

19.1.1. Extrinsic Plagiarism Detection

Extrinsic plagiarism detection is based on the pair-wise comparison of submissions. The goal is to detect similarities between documents and thus allow to judge whether one can be considered a copy of the other. Consequences and limitations depend on the context: If a copy within two submissions by different students is found, plagiarism detection cannot tell which one is the original. If a copy between a submission and an external document (such as an article in an online encyclopedia) is found, the case is more obvious. However, in both cases it must also be checked whether similarities between documents can also occur coincidentally. The latter is more likely, the more submissions to an item are available and the shorter these submissions are.

In general, the pair-wise comparison with other submissions means that a new submission is compared to all already existing submissions and a rating is produced that tells how suspicious the new submission is with respect to plagiarism. Notably, that does not only produce a plagiarism rating for the new submission, but can also change the rating for the existing ones. A typical process for extrinsic plagiarism detection as e. g. sketched by [10] consists of several steps: In the first step, some filter is used to select those documents from the set of existing submissions that should be compared with the new submission. Since the number of comparisons and thus the run-time grows exponentially in the number of submissions, any reduction is helpful. In the context of e-assessment, filtering is quite easy because submissions are clearly associated with assessment items and there is usually no need to compare submissions from different items with each other. In some contexts students are also allowed to make multiple submissions to an item and it can save some comparisons if only the latest submission per student is used or at least only those that show a remarkable delta to the previous one. On the other hand, filtering becomes more complex if also external sources (i. e. arbitrary documents from

the internet) are considered. In that case, heuristics are required to find a reasonable small but sufficiently large set of candidate documents that are used in the second step.

In the second step, a detailed comparison between the new submission and all older ones that passed the filter process is performed. The simplest possible comparisons are based on lexical or syntactic features that either split a document into n-grams (blocks of n characters or words) or based on some syntactic properties on the level of sentences or below. While character n-grams can be used on virtually any document, they are less suitable for submissions in some formal, domain-specific language that requires a large amount of fixed elements (e. g. XML or HTML tags). Word n-grams are only useful for natural language texts. Syntactic features require to use at least a tokenizer and potentially other tools like a parser for some formal language or a part-of-speech tagger for natural language. These assign labels to the tokens based on their syntactical meaning and thus allow to find plagiarism also in cases where minor changes on character level has been applied to a copy without changing the syntactical structure. More sophisticated comparison techniques can also look for semantic or structural features, potentially involving parsers for natural language or specialized dictionaries.

In the final third step, the individual results from the second step are combined to identify larger passages of a submission that are a copy from another one. The step can then compute a final rating based on the length of the suspicious passages divided by the total length of the submission and possibly include other factors into the rating such as some confidence score.

The simple approach is able to find any kind of literal plagiarism and also intelligent plagiarism that is based on text manipulation. It is not able to find plagiarism that is based on translation. For that type, a more sophisticated process is necessary that involves machine translation [219].

19.1.2. Intrinsic Plagiarism Detection

Another approach to tackle dishonesty in e-assessment is intrinsic plagiarism detection. That approach basically tries to identify lexical, syntactic, and semantic patterns in documents submitted by students. The assumption is that each student will produce an individual set of these characteristics and that he or she will do that consistently throughout a document. Consequently, a change in the characteristics of different passages within a document can be expected if they are copies from different sources. If the entire document is copied, that cannot be detected. In that case, authorship verification might be a better technique, that will be discussed in section 19.1.3 below.

The general approach to intrinsic plagiarism detection is somewhat simpler than the one for extrinsic plagiarism detection. In particular, there is no exponential growth in run-time with growing numbers of submissions, since every submission is analysed on its own. A typical process as e. g. sketched by [10] based on [218] is as follows: In a first step, the document is divided into smaller parts such as sections, paragraphs, or sentences. Then stylometric features of different types are calculated for each part. Lexical features include the frequency of characters or n-grams on the character level and various metrics on the word level. The latter include the average length of words or sentences, vocabulary

richness, frequency of (specific) words, and the occurrence of lexical errors. Syntactic features include sentence and phrase structure, characteristics based on part-of-speech tagging, and occurrences of syntactic errors. Even more sophisticated analyses can also include semantic and context-specific characteristics, including synonyms, homonyms, semantic dependencies, or context-specific or language-specific keywords. As already mentioned for the extrinsic plagiarism detection, syntactic features require to use a part-of-speech tagger for natural language, while lexical features require at least a tokenizer. Since formal languages allow less freedom in the lexical and structural design of an answer, intrinsic plagiarism detection is much harder to apply on documents in formal languages than in natural language. However, some features may also be detectable there, such as different styles in using line-breaks, indentations and in naming variables in programming assignments.

Based on the stylometric features, a style model can be constructed and outliers can be considered in a post-processing step [266]. The resulting model allows to identify passages of the document that are inconsistent with the style of the remaining document. Different to extrinsic plagiarism detection, where an identified copy as is quite sure sign of plagiarism (although it might be unclear who is the culprit), any passage flagged by intrinsic plagiarism detection still requires further manual inspection. In particular, it is not clear from intrinsic plagiarism detection, which of the passages of the document are actually created by the student and which are copied from other sources. That problem can be tackled by authorship verification.

19.1.3. Authorship Verification and Attribution

Authorship verification and attribution are more specific problems than intrinsic plagiarism detection, but basically use the same techniques in terms of stylometric features and style models [10]. The goal of authorship verification is to find out whether a given document (or part of it) was authored by a specific author, from whom some sample documents are known [171]. In authorship attribution, several candidate authors are known and a new document should be assigned to the correct author.

Authorship verification can be particularly useful in scenarios in which students make several submissions throughout some period of time (e. g. a term). If there is doubt whether one of these submissions (e. g. in the final exam) has indeed been made by the same person who made all the other submissions throughout the term, authorship verification techniques can be used to find out whether all submissions conform to the same style model. If the suspicious submission shows remarkable differences according to the style model, that can be used as an indication that the submission was created by a different person. The reasoning is thus basically the same as with intrinsic plagiarism detection, but with comparing several submissions to each other instead of comparing different passages within the same submission.

Authorship attribution goes a step further and can be particularly helpful in scenarios in which already some indication for plagiarism has been found. If for example extrinsic plagiarism detection has identified submissions from three different students as almost similar, authorship attribution can be used to find out who of the three students is the

most likely author of the original passage that was then copied and altered by or for the other two students. However, a requirement for successful authorship attribution is to have a set of sample submissions from each candidate where there is no doubt that these samples were indeed created by the respective students. Otherwise, authorship attribution can only produce indications that two submissions or passages have been authored by the same person, but it still remains unclear who that person is.

19.1.4. Involving additional data

The approaches discussed so far operate solely on the actual submissions made by students. They thus only consider material that has actively been prepared by the students for the purpose of assessment. That implies, that students could have made an effort to remove traces of plagiarism as good as they can. One approach to counter these efforts is to collect additional data within an e-assessment system, that are less easy to manipulate for students. In particular, biometric data can be used for that purpose. Some useful biometric data can be collected with standard sensors available on many computers. Typical options are face recognition (requires a camera), voice recognition (requires a microphone) and keystroke dynamics (requires a keyboard, but also works on mobile devices [314]), while other options like fingerprints or iris scans require special sensors and are thus not typically available [27].

The basic idea is that students provide a probe (an image, voice sample or an input sequence on the keyboard) during an enrollment phase. Feature extraction converts that probe into a set of vectors that represent characteristics of the probe that are considered to be individual for each student. The process is thus similar to the one used for the analysis of writing style. However, it cannot be manipulated that easy, because biometric features correspond to physical capabilities and characteristics of a person and thus cannot be changed deliberately.

Not all techniques are equally reliable and applicable. Face recognition produces excellent results [27], but cannot be used in scenarios in which students cannot be forced to use their camera. Keystroke dynamics produce good results [27, 314] but can only be used in a meaningful way in scenarios in which texts are produced in a way similar to the creation of the initial probe. Since keystroke dynamics can also be used for short inputs like passwords, length of the text is not necessarily a problem, but other context factors may be. In particular, keystroke dynamics for a student writing a text in English may be remarkably different from the keystroke dynamics for the same person writing a text in German or writing program code. Moreover, keystroke dynamics cannot be used at all in scenarios in which most of the inputs is created via mouse input. Finally, voice recognition produces less reliable results (i. e. in case of background noise) [27] and suffers even more than face recognition from the problem that students usually are not required to turn on their microphone for the actual purpose of the assessment.

A more basic approach on using additional data is thus to involve metadata like submission timestamps or similar events. Time series analysis can be applied to these data to look for correlations of events between the submissions from different participants [52]. These can be used as indicators that these participants worked closely together.

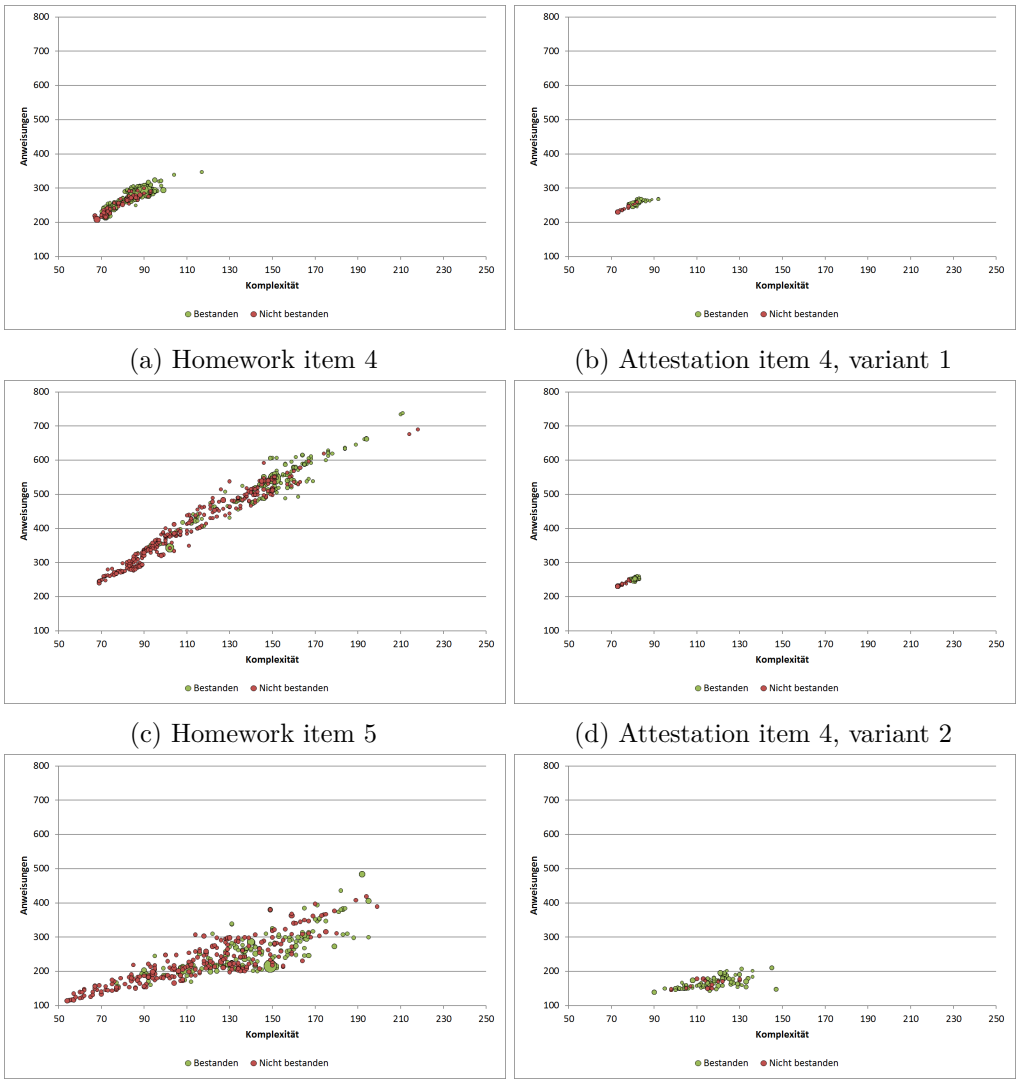
Hence, e-assessment systems in general offer ample opportunities to involve additional data that cannot be recorded in traditional assessments that easy. Both simple sensors and useful algorithms are available to involve that data into the authorship verification process. However, legal and ethical problems arise, if the specific sensor (i. e. microphone or camera) is not actually used during the assessment and is thus only required to record additional, personal data. Moreover, these is still the problem that the person sitting in front of a camera or microphone is not necessarily the person who is actually producing the inputs received by the system via keyboard and mouse.

19.2. Item Alignment and Answer Diversity

In several scenarios it might be interesting to use a set of items that are similar or evenly distributed over a range of values with respect to a specific property, such as difficulty or time required for answering the item. In particular, teachers might be interested in having several different items with the same properties or having a pair of items where one item is a simpler variant of the other item. Hence, items must be aligned to each other. Section 17.1 already explained how IICs and alike can be used to check the characteristics of items with respect to the competences they measure. Nevertheless, there might be other properties of interest.

For example, items on computer programming may ask students to create some piece of program code. So-called software metrics can then be used to measure the size and the complexity of the program code. Comparing the results of these measurements over a larger number of submissions per item as well as over submissions from different items can help to gain insights into the characteristics of the items. For example, the minimum or average size or complexity of a correct solution can be determined, which can help teachers to get a realistic estimation of the workload required by the item. Figure 19.1 shows examples of 2D-plots that visualize such information for some programming items [278].

Such visualisations allow for different comparisons. The left-hand column in figure 19.1 shows data from three different homework items. While the submissions to item 4 (figure 19.1a) are clustered closely together and thus show a low variance on both axes, homework items 5 and 6 (figures 19.1c and 19.1e) open up a much larger design space. Moreover, in homework item 6 there seems to be less correlation between the two axes than in homework item 5. Figures 19.1b and 19.1d show two attestation items that are supposed to be similar to each other and to be smaller versions of homework item 4. The 2D-plots provide indications that these assumptions are met. In particular, the plots for both items look almost similar, as they cover the same design space. Moreover, the covered space is smaller, but with almost the same center as homework item 4. Hence, they seem to give the students less room for variations or alternatives and can thus indeed be considered smaller. Similarly, figure 19.1f shows an attestation item that is supposed to be a smaller version of homework item 6. Indeed, the item seems to be much smaller with the same reasoning as above. However, it can also be seen that the average value on the x-axis seems to be similar for both items, while the attestation item shows a



(a) Homework item 4

(b) Attestation item 4, variant 1

(c) Homework item 5

(d) Attestation item 4, variant 2

(e) Homework item 6

(f) Attestation item 6, variant 1

Figure 19.1.: 2D-plots for size and complexity of submissions to programming items [278]. The axes provide two different software metrics, while the size of the circles indicates the number of submissions with the respective characteristics. Green circles are passed submissions, while red circles are failed submissions.

lower average value on the y-axis. That means that the item is of similar complexity, but of lower size, which is probably indeed the intended relation between the items.

Another similar metric can be found in literature as "diversity rate" that can be applied more generally to complex artifacts and not just to program code. In that approach, a complex answer (such as a diagram) is divided into smaller components. Similar components across answers from different students are grouped into blocks. The diversity rate is then calculated by dividing the number of component groups by the total number of components. Hence, the diversity rate is low if answers are similar to each other and high otherwise. This can help item authors to judge the quality of their items, as answers for items providing e. g. similar scenarios should have the same rate [29].

19.3. Meta-Data Analysis

Most of the opportunities for item and answer analysis that have been discussed in the current chapter so far are actually not limited to e-assessments. Although it might require some additional effort to transfer non-digital submissions into the right format, most techniques for plagiarism detection or item alignment analysis can also be applied to submission from traditional exams. The only exception so far was the additional data that could be involved in plagiarism detection in terms of keystroke dynamics and alike. However, there is more additional data that can be collected by an e-assessment system and that can be relevant for further analysis. Since the additional data is not part of the actual submission (i. e. it is not created intentionally by the students) but can be used to learn more about the submission or an assessment item, is is called meta-data in the following. In general, there are two different types of meta-data: Data that is collected while a student is working on an item and data that is collected during or after the grading process.

19.3.1. Student Data

Even in the case of very simple assessment items, students will need some time to create or prepare their answer and they will perform at least a very small amount of interactions with the e-assessment system (e. g. at least tick a checkbox). In more complex items, much more events can be recorded, e. g. frequent changes between phases in which students type and phases in which they (presumably) think, or actions with which they partially delete what they have entered before. Although these events are not relevant for the actual grading, they can help teachers to understand the difficulty of items or analyse the way in which students work on these items.

For example, e-assessment system that are used in formative assessments may provide feedback for wrong answers and offer students the opportunity to submit another answer. The time between the first submission and the second submission can at least partially be considered as "feedback study time". Depending on the actual design of the student interface, it may also be possible to measure that time exactly, if students need to click some button to dismiss the feedback and before they can enter a new answer. This kind

of meta-data collection is not easily possible without e-assessment systems, but can be very valuable in education research (e. g. [67, 195]).

Depending on the domain of the assessment, submissions may be complex artifacts that answer several sub-tasks at once as e. g. program code in assessment items on computer programming can do. E-assessment systems may allow students to make an unlimited number of submissions and thus allow them to improve their answer step-by-step by solving one sub-task after another. Besides that fact that students may receive feedback between these steps and thus may spent some time studying that feedback as discussed above, one can also observe the number of solved sub-tasks in each submission. Different to a paper-based assessment, in which it is not easy to see how much time a student spent on each sub-task, that allows not only to measure time, but also makes it visible if a student made more than one try to solve a sub-task. While that is typically irrelevant for the final grade, it can again help teachers to understand the way how students work on the assessment items and also help to judge the difficulty of items or the underlying concepts (see e g. [150]).

19.3.2. Grading Data

In part III of the current publication, different mechanisms for grading submissions have been discussed. Some of these mechanisms involve complex procedures like applying large rule-sets or executing test cases. For these procedures, data can be collected in terms of e. g. time needed for grading an individual submission or the frequency each rule matched on a larger set of submissions. While meta-data on individual submissions might be used for direct feedback, meta-data on grading can also be used to analyse the items.

In items on computer programming, measuring execution time is quite simple, as the program code submitted by students is executed anyway. The measurement can be performed in time units or in program steps, which both have individual advantages and disadvantages. A measurement in time units requires that the tests for different submissions are performed always on the same or at least an identical system and always under the same conditions. Deviations in system load or system performance inevitably lead to different results for identical submissions, which makes the results less meaningful. A measurement in program steps is not subject to this risk, but is much more difficult to interpret, since individual program steps do not have to be equivalent and no general statement can be made as to whether a solution with fewer steps is better than a solution with more steps.

Besides providing feedback to students, the meta-data on execution time can in particular be used to determine the quality of the test cases. For example, a study on the execution times for test cases from six different exercises revealed that one of the exercises required remarkably more time for grading [267]. In fact, that deviation could be traced back to unfavorably formulated test cases, which led to long run times without improving the actual test quality. As a consequence, the test cases could be improved, which in turn led to faster feedback for the students. Hence, checking and improving the

quality of individual items cannot only improve the quality of grades or feedback, but also the quality of the general system behaviour.

20. Case Studies

This chapter presents two cases in which assessment data was used in various combination and for different purposes. Both case studies are related to the e-assessment system JACK, but the actual topics and goals of each study are independent from that system. The aim of the chapter is to demonstrate and explain examples on how to integrate assessment data into a didactical analysis.

20.1. Case 1: Learning Effort and Final Grade Prediction

The first case study demonstrates how to use aggregated data from an e-assessment system in conjunction with external data from final exams to gain insights into the relation between learning effort and outcome. The study thus focuses on data on people, but it also produces data that can be used to judge assessment items. It summarizes results from two earlier publications [186, 185] and a bachelor's thesis by Maria Berski [31], that apply the same methods to data from two different courses (an introductory course on statistics and an introductory course on programming). In both cases, data has been extracted from the e-assessment system JACK once for the purpose of a research study, but the analysis based on that data can easily be repeated frequently, e. g. every term. Data sources in both courses are exercises offered continuously during the term, mandatory attestations that have to be taken frequently during the term, and a final exam at the end of the term. Students can gain up to 100 credit point for each exercise and attestation. Final exam grades are translated as "100" (best grade), "200", "300", "400" and "500" (failed) in both studies. However, the methods of data analysis used in the studies are also applicable to data that differs in detail, but follows the same structure.

The studies made use of several statistical analyses to answer the question whether there is a significant correlation between the learning effort of students during the term and the results they achieve in a final exam. In particular, one can make several assumptions related to learning effort: There may be a correlation between the mere effort (e. g. the number ob submissions to exercises) and the final grade or there may be a correlation between the exercise or attestation results and the final grade. Moreover, a typical observation of teachers is that some students work continuously during the term, while others only work hard short before the exam. Hence, it could be possible to prove or disprove these observations based on assessment data.

20.1.1. Learning Effort

Ordinal logistic regression was used to analyse the relation between learning effort and final grade. Three different regressions were used:

- Final grades on number of exercise submissions.
- Final grades on total sum of exercise credit.
- Final grades on total sum of attestation credit.

The introductory course on statistics had 493 participants and included 151 different exercises, resulting in a total amount of 131,426 individual exercise submissions. Due to these large numbers, a strongly significant prediction model could be reached, with $p < 10^{-13}$ for the first regression and $p < 10^{-16}$ in the other cases (Wald-test). Figure 20.1 shows the predicted probabilities to obtain each grade against the number of the exercise submissions. For example, the red long-dashed curve on the left-hand side is the estimated probability to fail the exam. The probability is about 55% if a student has no submissions at all and it decreases monotonically in the number of submissions. On the other hand, the probability for the best grade is increasing and the modes of the three other curves are ordered in increasing order of the grades. Figure 20.2 shows very similar curves for the predicted probabilities to obtain each grade against the total sum of exercise credit. The modes are ordered in the same way and maximum probabilities are slightly higher than in the previous figure. In conclusion, it is significantly worth both to invest time in many exercise submissions and to attain more points in these. Finally, figure 20.3 shows another set of quite similar curves for the predicted probabilities to obtain each grade against the percentage of obtained attestation credit.

For the introductory course on programming, 242 participants produced a total amount of 17,108 submissions in 66 exercises. Since these numbers are substantially lower, no significant prediction model could be achieved with the first regression. However, using the total sum of exercise credit produced a very significant model ($p < 10^{-3}$) and for the attestation the model is also strongly significant ($p < 10^{-8}$). Figure 20.4 shows the predicted probabilities to obtain each grade against the total sum of exercise credit. The curves are much flatter than the ones for the statistics course, but nevertheless show the same general characteristics. The curves for the predicted probabilities to obtain each grade against the total sum of attestation credit in figure 20.5 look slightly different, because the x-axis starts at 330 points of attestation credit, which is the minimum credit to take part in the final exam in that course. Nevertheless, the characteristics of the curves are similar to the previous one.

Students may potentially use such prediction models directly to compare their effort with the one that is most likely required for the desired outcome. Notably, the prediction models only use aggregated data and publishing these models thus involves no serious privacy issues. Students can then do the analysis of their personal effort on their own and thus do not expose additional personal data.

The fact the the curves for the programming course are flatter than the one for the statistics course allows for interpretation and requires further investigations. One possible

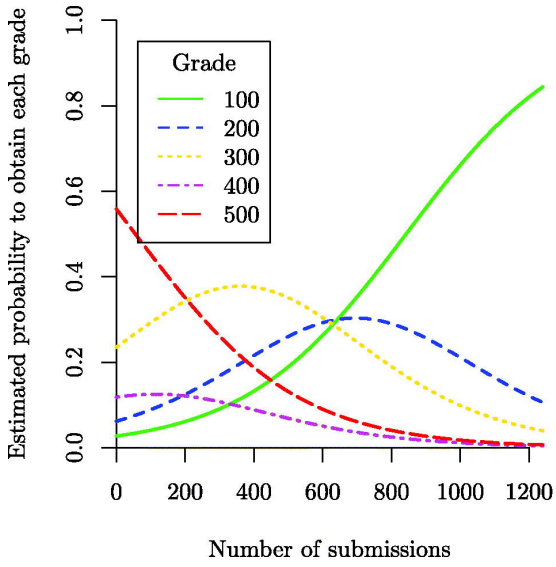


Figure 20.1.: Logit regression grade on submissions. The figure shows the predicted probabilities to obtain each grade against the number of the submissions for the statistics course. [186]

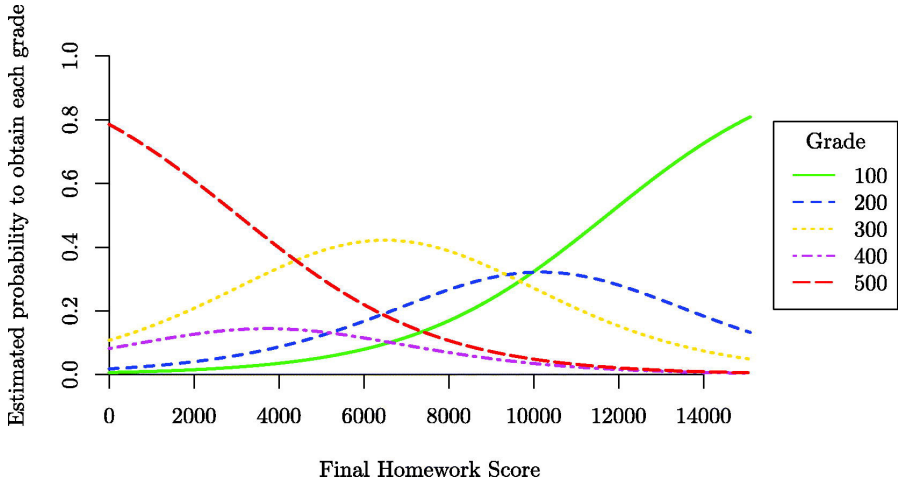


Figure 20.2.: Logit regression grade on exercise credit. The figure shows the predicted probabilities to obtain each grade against the total sum of exercise credit for the statistics course. [186]

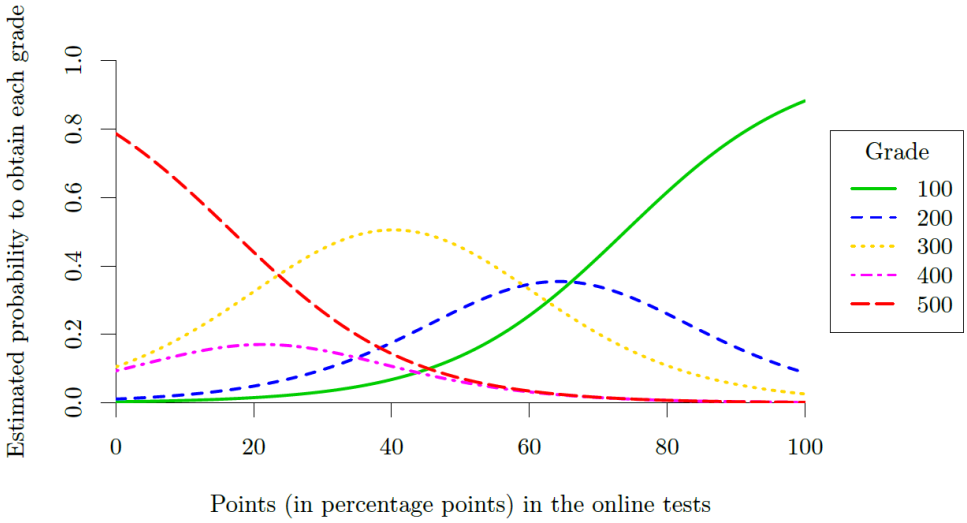


Figure 20.3.: Logit regression grade on attestation credit. The figure shows the predicted probabilities to obtain each grade against the percentage of obtained attestation credit for the statistics course. [186, Supplementary material]

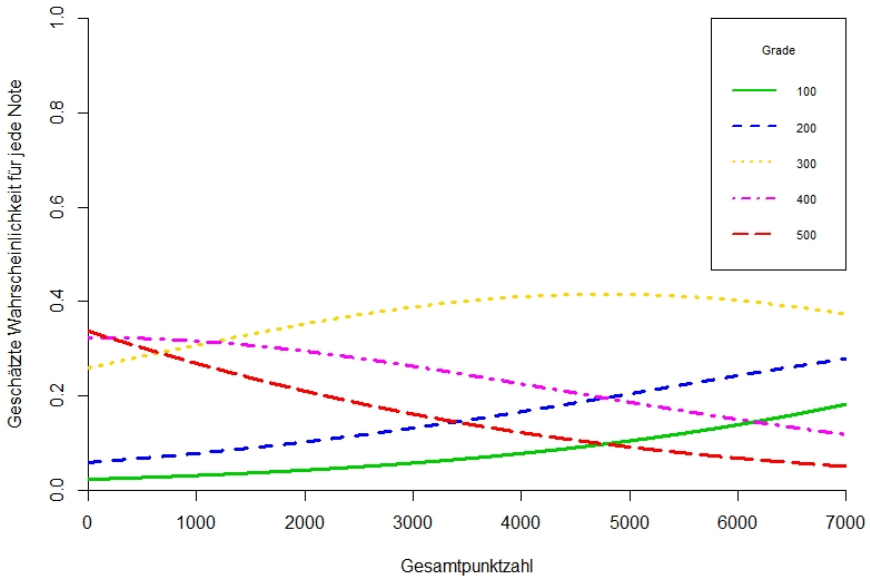


Figure 20.4.: Logit regression grade on exercise credit. The figure shows the predicted probabilities to obtain each grade against the total sum of exercise credit for the programming course. [31]

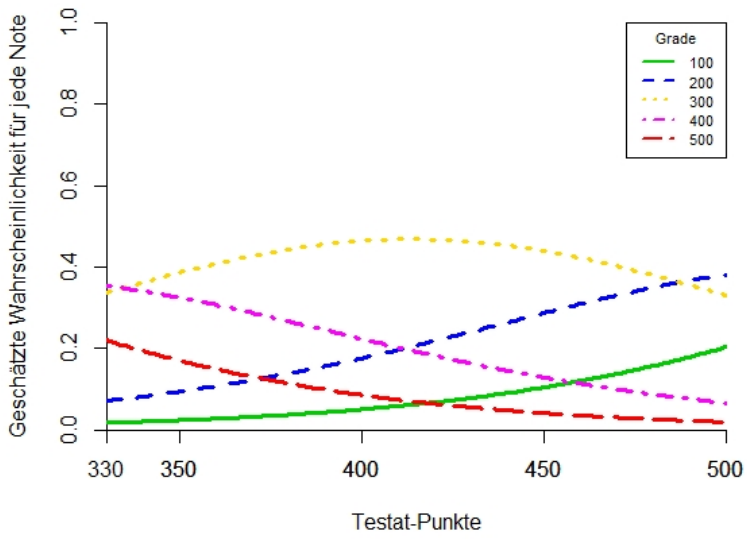


Figure 20.5.: Logit regression grade on attestation credit. The figure shows the predicted probabilities to obtain each grade against the total sum of attestation credit for the programming course. [31]

explanation is that in the programming course a minimum of 330 points of attestation credit was required to take part in the final exam. Hence, the group of students taking part in that exam is more homogeneous in terms of exercise and attestation performance and hence other factors become of larger importance. Another possible interpretation is that the final exam in the statistics course is more aligned with the exercises and attestation and thus the results can be predicted more precisely. If that is the case, the data analysis does not only produce useful data for students to judge their own learning effort, but it also helps teachers to check whether their exercises and attestations are suitable to prepare their students for the final exam.

20.1.2. Learning Progress

The previous section demonstrated the possibility to predict exam results from exercise and attestation results. While this is helpful to judge individual learning effort, some open questions remain. In particular, students probably do not make any substantial learning progress, if they repeatedly submit several correct submissions to a single exercise. It nevertheless increases their total sum of credit and thus the predicted probability for a better grade. Moreover, it might be interesting to see whether there are differences in progress over time between students with higher and lower grades.

An estimation of the individual learning progress can be made by tracking each student's sum of points of the last submissions to every exercise. This sum increases

with every exercise that is solved for the first time or every attempt that is better than a previous one, but decreases with unsuccessful attempts that follow a more successful one. Hence, if a student got 100 points in an exercise at the first attempt but only 50 points at the second, the sum decreases by 50 points from the first to the second try. These sums can then be averaged within groups of students that received the same grade in the final exam. The changes in these values over time can then be plotted to depict the learning progress of the different groups.

Figure 20.6 shows the average progression of the student score for each grade group in the statistics course throughout the term. Additionally, the solid black line describes the average progression for all students. Vertical lines represent the four exam dates for that course. Since there were 151 exercises offered and each is worth at most 100 points, the maximum reachable credit would be 15,100, but even the students with the best grades only reached about a half of that value on average. The general tendency in the curves is, that the better the final grades, the better the students have developed according to their score. Notably, the curves for students with a moderate exam ("400") and those who failed ("500") are similar at the beginning of the term. However, the "moderate" students made substantial progress in July, while the failing students barely improved. Hence, both groups are moderately successful in the beginning, but the passing students improve shortly prior to the exams.

A similar observation can be made in figure 20.7, that shows the respective curves for the programming course. Only the vertical line on the right-hand side represents an exam data in this case, while the others represent the dates of the attestations. A remarkable difference to the statistics course is the fact that the curve for moderate students ("400") is similar to the curve for satisfactory students ("300") for about a half of the term and then is even above that curve. A proper interpretation of that fact requires further investigations.

Several other observations can be made that may lead to additional hypotheses about learning behaviour and progress. In particular, both graphs show curves for different grades that are closes to each other in the beginning and then split remarkably shortly before an exam or attestation date. Since all curves rely on aggregated data, they can be used to demonstrate the effects of steady learning progress without causing much data security issues by revealing individual, sensitive data. As in the previous section, individual students can use that data to compare their individual progress with the progress of their peer group.

20.1.3. Exercise Relevance

Decision trees have been tested in both courses as an additional means for predicting final exam results from previous exercise and attestation results. In both courses, a correct prediction could be made in 85% of the cases [185, 31]. An interesting side-product of using decision tress (e. g. with the CART algorithm) is the fact that a relative importance of the available variables within the model needs to be computed (see section 19.2).

For the statistics course, the most important variables turned out to be the attestation credit in the last two (of five) attestations and the sum of exercise credit until the third

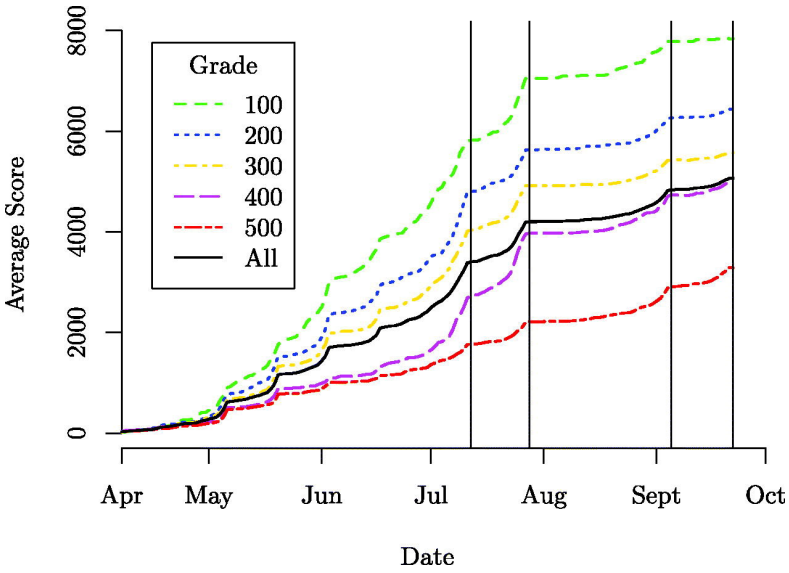


Figure 20.6.: Learning progress over time. The figure shows the average sum of exercise credit for the statistics course grouped by students who received the same grade in the final exam. [186]

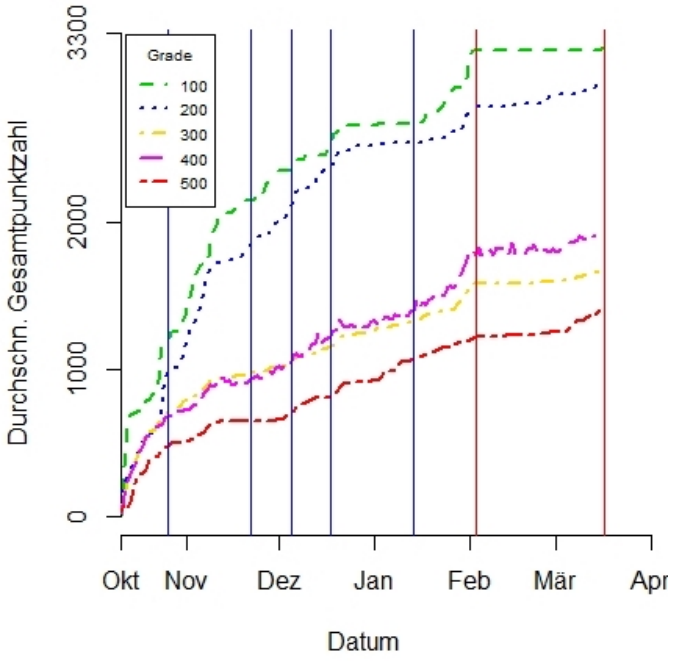


Figure 20.7.: Learning progress over time. The figure shows the average sum of exercise credit for the programming course grouped by students who received the same grade in the final exam. [31]

attestation. This is somewhat reasonable with the following interpretation: If one assumes that the later attestations require more competences, they should be more aligned with the final exam than the earlier ones. In turn, individual results from earlier attestations should be of low importance, while a general lack of learning progress in the first half of the term (as expressed by a low total sum of exercise credit) makes it hard to follow the second half of the course. Consequently, there seems to be a good alignment between the exercises and the attestations on the one hand and the final exam on the other hand. Additionally, there seems to be a good balance in the relevance of the exercises in the first half of the term and the attestations of the second half of the term, as both are relevant for predicting the final exam result.

Different results could be reported for the programming course. In that course, the attestations were in general of higher importance than any exercise score. Moreover, the attestations from the middle of the term were much more important than the ones from the last third. Hence, the exam seems to be much more aligned with material from the middle of the course. Moreover, the dominance of attestation results possibly reflects the fact that a minimum sum of attestation credit is required to take part in the final exam. In particular, students who practice hard in the exercises but fail in the attestations are not admitted to take the final exam, while there might well be students that pass the attestations without any substantial work on the exercises (e. g. due to previous knowledge from school). Further investigations are necessary on these hypotheses and results can help teachers to improve their course and to offer more relevant exercises. In turn, the results from the statistics course prove that attestation credit may be indeed a relevant admission criterion to make sure that only students who have a good chance to pass are allowed to take the final exam.

20.2. Case 2: IRT on Programming Items

The second case study deals with a combination of domain-specific item handling for items on computer programming on the one hand, and classical item response theory (IRT) on the other hand. The goal of using that combination is to gather insights into the difficulty of programming constructs and to use that for competency measurement. The case studies thus deals with data on items as well as with data on people. Data has been used once for a research study, but the results can easily be re-used for continuous use for adaptivity or competency measurement.

The general procedure in the research study was as follows: In a first step, items (in terms of IRT) are defined in form of arbitrarily complex structures that are either present or absent in the source code artifacts submitted by the students in response to an assessment items. In the second step, these items are formalized to allow for automated detection by means of domain-specific item handling, similar as they are used for direct feedback generation. The result of the automated detection is a rating of occurrence or non-occurrence for each item in each artifact. In the third step, the item set is checked for homogeneity and potential violations are solved by removing items. In the fourth step, methods of classical IRT can be applied to the remaining homogeneous item set. The

approach thus integrates the automated analysis of artifacts from e-assessment systems with a psychometric analysis.

20.2.1. Rule-based Static Code Analysis

The rule-based static code analysis applied in the approach is based on TGraphs, which were invented for graph-based modeling [77]. They are based on a schema definition, describing legal graph structures and thus representing the structure of the underlying grammar. Queries on TGraphs can be expressed using a query language named GReQL [32]. This language is somewhat similar to SQL and thus well suited to implement rule-based checks: Queries in this language allow searching for elements of particular type, that are connected in a specific way, and own specific attributes. It allows to formulate graph queries of the following types:

- Existence of graph elements based on their type;
- Existence of graph elements based on the value of their attributes;
- Existence of tuples of elements based on the connections between them.

Executing queries of these types via the graph query engine returns either a list of elements matching a specified condition, or an empty list. It is also possible to combine different graph queries by logical operators to allow for alternatives. For example, all pieces of code containing a `for`-loop or a `while`-loop (or both) are found trivially by combining several queries via the logical OR.

20.2.2. Code Analysis with Item-Response Theory

Usually, in item response theory, a test with several evaluated items is conducted on a particular population. Afterward, the items are checked, and parameters are calculated for the items and the individuals of the test. As programming, in general, is a complex task and refers to several cognitive (and, therefore, latent) processes, the item response theory is applicable. In a classic setting, small programming tasks on a particular concept or code element are provided to the participants of a programming or coding assessment. However, if the coding ability – in the sense of a competency – must be evaluated, the tasks have to be more complex. The complexity causes difficulties in separating latent constructs and in assigning an ability to a particular task. Nevertheless, the resulting program code contains the responses to those tasks. This is the reason why the code items can be assumed to be the tasks posed to the participants, even though they were not. Notably, the tasks assumed here are primarily about using a particular code construct in a syntactically correct way. They are not about using code in a semantically meaningful way in the first place.

After all the items are rated with either “yes” (1) or “no” (0), the non-parametric test on homogeneity is applied to the resulting matrix. The output of the test is a set of item pairs that violate the homogeneity assumption. The number of violations orders the items. Afterwards, one item of the list after another is eliminated from the original

item set until the list of violations is empty. This process is repeated until there are no violations anymore. The remaining subset is homogeneous. However, this method only finds one possible subset. There might be others, but as the probabilistic approach is very resource consuming, this limitation is accepted.

20.2.3. Experiment

In the research study a small experiment dealing with the nesting of control structures was conducted. The aim of the experiment is to reveal item characteristics for different kinds of nestings of control structures. Based on that, learners can be classified according to the type of nestings they use. The data was collected in a voluntary, preliminary course for the first-year students studying Computer Science (CS).

The students were asked to implement either a "Mastermind" game, a tool for managing results from a sports tournament (e. g., a football league), or a version of the dice game "Yahtzee". The assumption was that these three projects have different difficulty levels. The projects are described in a way that the implementation can be as complex as the participant can do it. All three projects have in common that they encourage using nested control structures. Nevertheless, all projects can be solved without using nesting and also no project's complexity is dominated by the requirement of nesting of control structures.

For the detailed analysis, static analysis rules were used that detect single and nested control structures (i. e. `if`-, `for`- and `while`-statements). To conform to the Rasch model, all rules need to detect independent structures. Hence, single control structures are only considered if they are isolated, while nested ones count in separate categories.

20.2.4. Results and Interpretation

In total, the program code produced by 350 students was investigated. All provided program code was syntactically correct and thus runnable. In order to find homogeneous item sets within the data set, all combinations of items containing three to five items (because of the limited number of participants) were checked using the non-parametric tests. Three homogeneous subgroups could be identified that have a semantic correlation of the items: The first group of items consist of all combined elements related to the `for`-statement (hence, a `for`-statement within a `while`-statement, a `while`-statement within a `for`-statement, an `for`-statement within an `if`-statement, an `if`-statement within a `for`-statement, and a `for`-statement within a `for`-statement). The second group contains all items related to combined conditionals, i. e. combinations of conditions in `while`-loops and conditional statements (hence, an `if`-statement within a `while`-statement, a `while`-statement within an `if`-statement, an `if`-statement within an `if`-statement, and a `while`-statement within a `while`-statement). The third group contains all three isolated statements.

For all three groups, a simple Rasch model is calculated to get the person scores. According to a Goodness-of-Fit test [25] the model fit is satisfactory. Figure 20.8 presents the item characteristic curves (ICCs) of the first group. For better readability, the item

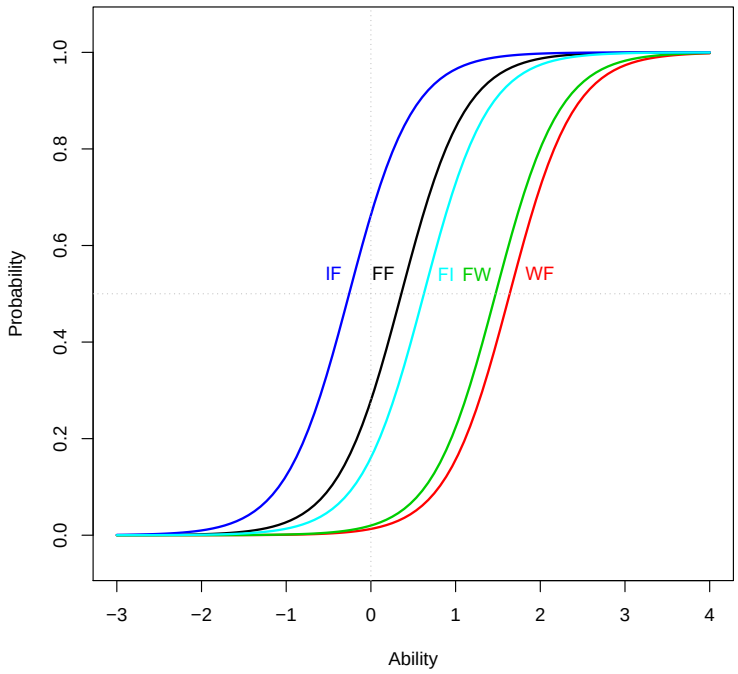


Figure 20.8.: Item characteristic curves for the item group containing items "`if`-statement within a `for`-statement" (IF), "`for`-statement within a `for`-statement" (FF), "`for`-statement within an `if`-statement" (FI), "`for`-statement within a `while`-statement" (FW), and "`while`-statement within a `for`-statement" (WF).

parameters are normalized, to sum up to 0. The item "`if`-statement within a `for`-statement" is the most left one (difficulty: -0.26), followed by the items "`for`-statement within a `for`-statement" (difficulty: 0.36) and "`for`-statement within an `if`-statement" (difficulty: 0.63) where the curves are very close together. Items "`for`-statement within a `while`-statement" (difficulty: 1.47) and "`while`-statement within a `for`-statement" (difficulty: 1.64) are the items rated most difficult are close together as well. Similar curves can be obtained for the other groups.

The data allows to draw conclusions on the items (in terms of IRT) and from those to come to insights into the structure of the assessment items. The data from Figure 20.8 allows to conclude that the implementation of a conditional statement within a `for`-statement is the combination that is preferred by the students and that is hence probably the most suitable one in the context of the tasks. The next pair of two items have in common that a loop is nested in another structural element. The last pair of items contains the cases where two different types of loops have to be combined. That allows to argue that an assessment item requiring a conditional statement within a `for`-statement

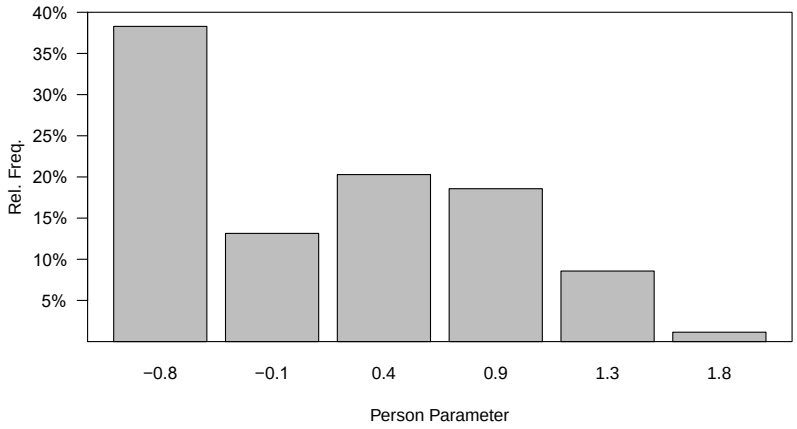


Figure 20.9.: Person parameter distribution for the group of items related to the `for`-statement.

is truly different from an assessment item that requires the nesting of two types of loops. In turn, it can be argued based on that data that it is not that relevant in which way two different types of loops are nested.

The data also allows to draw conclusions on the persons, since person parameters corresponding to the item parameters can be computed. These reflect the ability a person has regarding the latent construct. The distribution for the six possible person parameters for the combinations of `for`-statements can be seen in Figure 20.9. The person parameter -0.8 has the biggest proportion of 38%. The last person parameter is only reached by a few participants while the middle four ones are reached by 10% to 20% each. The median of the person parameters is at -0.1 while the mean value is 0.08. The distribution has its peak on the left side. So, most participants only use simple items in this category and have therefore a low ability. The item rated as easy is a conditional statement inside a `for`-statement, which is a quite common construct in programming. So, the majority of the participants only used this construct and no others. In the end, the characteristic of the participants can be found in the variety of combinations they use. While those with low ability stick to only one or two combinations, the ones with high ability use a broad variety of control structures. A more detailed analysis of possible correlations with other factors (like previous knowledge or experience) is of course necessary to come to better conclusions on what the actual latent competency behind the data is.

21. Results

The previous chapters had the goal to provide an overview on the different kinds of data that are produced in and by e-assessment systems as well as the different mechanisms used to process them. The chapters were roughly structured by the goal of data processing, starting from competency measurement as the goal most directly linked to educational assessment, followed by other analyses focused on learner achievement and performance, finally ending with analyses that help to judge and improve the integrity and quality of assessments. The overview allows to summarize some results and draw connections to the earlier parts of the current publication.

21.1. Contributions to Integrated E-Assessment

Using the data produced in and by e-assessment systems as good as possible is an important aspect of a proper integration of e-assessments into a larger context. E-assessment systems do not operate in an isolated manner, in which they receive some input by students and only produce grades and feedback related to these submissions. Instead, the various types of data form several interfaces via that e-assessment systems communicate with the surrounding context: Data from previous assessments or enrollment systems can be used to verify authorship and fight plagiarism; data collected across several submissions by a single student can be used to create a competence profile of that student; data collected across several students for a single item can be used to assess and improve the quality of assessment items; and data collected across multiple students and assessment items can be used to predict success rates. All of these goals relate to the context of an assessment, in which a single assessment is not an isolated event, but an event with consequences on the future development of persons, learning materials and assessment items.

At the same time, any data processing related to these goals is subject to legal and ethical issues. Although e-assessments are integrated into a context, the main purpose of an e-assessment system is still to conduct assessments. Any data processing that goes beyond that purpose needs a proper justification and transparency about the ways in which data is stored, combined, processed and forwarded to persons other than the one who is responsible for the assessment.

Some of the methods and goals for processing e-assessment data that were discussed in the previous chapters directly relate to some of the states in the assessment process models presented and discussed in part I of this publication. In particular, state “Outcome reviewed” for alpha “Test Items” (see table 3.1) and state “Evaluated” for alpha “Test” (see table 3.2) both refer to a deeper analysis of assessment data. Both state explicitly mention the application of suitable metrics for test items and test, as they have been

discussed for example for IRT in section 17.1. Similarly, item alignment as discussed in section 19.2 can be a suitable technique to identify the causes of unwanted characteristics as mentioned in the last state for alpha "Test Items". Hence, the previous chapters provide a rich tool set that can help to reach the checkpoints for the states in the kernel for educational assessments and fill the related activity spaces with actual activities and techniques.

At the same time, some of the techniques from the previous chapters demonstrate why domain-specific item handling as discussed in part III is so important: Some methods for learning analytics and outcome prediction rely on detailed and fine-grained outcomes from many exercises, and competency measurement for complex artifacts via item response theory is even only possible if domain-specific techniques are available that can decompose such artifacts into smaller, meaningful units. Hence, the availability of domain-specific techniques that allow for the fine-grained automated evaluation of test item responses allows for much richer data processing and thus more applications of assessment data analyses. In turn, domain-specific item generation can benefit from detailed data processing, i. e. in case of adaptive systems where results from competency measurement or item alignment can help to generate items with a specific difficulty.

The connections to part II of this publication are less strong. This is not surprising, if the various types of data are considered as interfaces between e-assessment systems and their context. Many questions that can be answered by learning analytics or item and answer analysis can be handled outside and actual e-assessment system, if the data is available for export. In particular, learning analytics often does not only rely on data from e-assessment systems but also includes data from learning management systems and other sources. Hence, e-assessment systems are not the systems in which the related data processing actually happens and thus no components specific for that purpose were identified in part II. The only remarkable exception is the use case of adaptive e-assessment, where results from competency measurement has a direct influence on the assessment. Consequently, student models that reflect the competence profile of a student had been identified as one of the storing components in section 6.1.5 and directly relate to data handling. However, several important legal and even ethical issues related to data processing cannot be handled solely on the level of plain data, but require appropriate measures for authorization or pseudonymization on the system level. And even more trivially, data can only be analysed if system design has provided the necessary means to collect data, which is specifically important in the case of meta-data on the submission creation process that require a fine-grained log of all student activities.

21.2. Contributions beyond the Scope of this Publication

Many of the techniques discussed in the previous chapters are not limited to the narrow scope of e-assessment systems. Actually, most techniques exist in that broader scope anyway and are just applied to e-assessments, but have not been invented specifically for that purpose. For example, competency measurement via item response theory is not limited to the use of computers at all, but can also be performed with paper based

assessments. The same is true for the problem of plagiarism detection and authorship verification, which is of course also relevant in any assessment that is conducted without using an e-assessment system. However, most aspects of e-assessment data at least benefit from the fact that e-assessment systems can record and process data more easily than it is possible in traditional, non-electronic scenarios.

Nevertheless, most goals and methods for data processing can be used in a much wider context. As already mentioned above, learning analytics usually also draws data from other data sources and hence the summaries on different techniques apply to scenarios without e-assessment data as well. Moreover, the same techniques cannot only be used with different input data, but can also be used for prediction goals other than assessment outcomes. Similarly, the techniques used for item alignment and meta-data analysis can also be used to compare artifacts in other context, e. g. to find out whether activities that are supposed to be of similar difficulty (like cooking different meals following given receipts or assembling different pieces of furniture following given instructions) are indeed similar. Actually, the only difference would be that people are not performing these activities for the purpose of assessing their skills or competences, but for the purpose of using the resulting product. However, the number of steps to be performed, the time between steps, the number of errors made and so on can be recorded and analysed in these scenarios exactly as it can be done in e-assessments.

In any case, the discussion on legal and ethical issues also applies to a much wider context than the one of e-assessments, since they are based on very fundamental laws and considerations. It has become clear that there is no way to use data just because it is available or can be made available. However, the previous chapters also demonstrated that there are indeed several good reasons for collecting specific data and that it is possible to precisely name the purpose of processing that data. These discussions thus help to argue why certain data is needed and at the same time help to find out which data possibly needs not to be collected at all. The discussion on data produced by and in e-assessment systems can thus contribute to a much wider discussion on the general use of fine-grained personal data in higher education and beyond.

Part V.

Conclusions

22. An Integrated View on E-Assessment

The previous parts of this publication tackled four major aspects of e-assessment. While all of these parts have their own state-of-the-art and their own challenges, the ultimate goal of this publication was to create an integrated view on e-assessment. As we have seen in all four parts, there are many connections between these parts and there are also many connections between e-assessment and the context in which it takes place.

22.1. Integration within E-Assessment

The first connection that we have explored is the integration (or the lack thereof) of software components for e-assessment and the educational assessment process. The latter forms a kind of a map of steps that have to be taken and thus implicitly names the features software components must offer. In turn, software designers can derive states and activities from the kernel of educational assessment and thus come to informed decisions on what functional interfaces they have to create for their components. The component catalogue showed some remarkable matches between software components and parts of the process, but also some blind spots where there are no commonly used components available yet. In turn, we have seen that the availability of software components can also have an influence on the processes, if teachers cannot freely decide how to conduct their assessment because certain software components are not available. Thus we can conclude that there is a clear need for a close integration of educational processes and software components, where the educational processes should be the driver for the design of component features and interfaces. Unfortunately, the current situation also knows scenarios in which existing components actually limit the execution of educational processes.

The second connection that we encountered is the realization of domain-specific features by specialized software components or domain-specific extensions to existing components. We could identify some cases in which e-assessment can only be used in a meaningful way if a system is able to handle domain-specific data formats, provide domain-specific editors or run domain-specific algorithms for item generation or answer evaluation. In turn, we have seen that there are already some commonly used components that are specifically designed for the related tasks and that can thus be used in multiple instances for different domains. In addition, we have seen several patterns (e. g. for the integration of grading components) that help to obey the specific requirements of a particular domain. Finally, the modular software design is an important prerequisite to be able to add domain-specific services that offer no actual e-assessment capabilities, but increase the usability of a system for that domain. Thus we can conclude that a modular software design with well-defined interfaces eases the integration of domain-specific features into e-assessment

systems and thus makes e-assessment available to a broader group of users. In turn, various domains can offer existing tools for integration and thus extend the feature set and scope of existing e-assessment systems.

The third connection that we came across was somewhat weaker, but not less important. It concerns the relation between educational processes and domain-specific data sources and tools. While the elements in the kernel of educational assessment specify what to do to prepare, conduct or grade an assessment, the domain-specific data-formats and tools specify how to do it in a particular case. This is a quite valuable connection, since the availability of e. g. tools for automated item generation or grading can have a remarkable influence on the time each step in the educational process consumes, while the absence of appropriate tools calls for the manual execution of the respective process steps. In turn, the decisions made during the planning of an assessment drive the requirements for domain-specific tools for e. g. user input or answer representation. At the same time, the kernel for educational assessment is general enough to cover assessments from any domain and it also allows for domain-specific extensions. Thus we can conclude that the need for or the availability of domain-specific tools should be considered during the creation of an assessment process. In turn, mapping an existing process to a specific domain can help to identify the domain-specific tools that need to be in place and thus require integration into an e-assessment system.

The fourth connection that we have identified also relates to assessment processes. Several states in the kernel of educational assessment refer to a deep and detailed analysis of assessment data and thus advocate for the integration of analytical capabilities into processes and systems. Similar to the previous case, the kernel specifies what to do in order the check and improve that quality of an assessment, while the integration of actual analysis techniques specifies how to do it. The specific value of e-assessment in that case is the fact that all data is already available electronically and thus analysis is much easier than in traditional assessments. Hence we can conclude, that e-assessment itself is a great driver for the integration of advanced analytics into educational processes with benefits for both sides: Processes provide the questions to be answered by the data, and automated continuous data analysis allows for much more agile assessment processes, including (but not limited to) adaptive and personalized assessments.

The fifth connection that we could see concerned the integration between domain-specific tools and data analysis. This is a very interesting connection, because on the first glance data analysis starts when all items are graded and thus works on the abstract level of aggregated data or general psychometric models, while domain-specific tools have finished their job when an item is graded and thus work on a very concrete level of individual items and answers. However, we have seen that some methods for learning analytics and outcome prediction rely on detailed and fine-grained outcomes from many exercises, and competency measurement for complex artifacts via item response theory is even only possible if domain-specific techniques are available that can decompose such artifacts into smaller, meaningful units. In turn, domain-specific item generation can benefit from detailed data processing, i. e. in case of adaptive systems where results from competency measurement or item alignment can help to generate items with a specific difficulty. Hence we can conclude that the integration of results from domain-specific

grading into general assessment analysis allows for a deeper analysis, while the integration of analysis results into domain-specific tools (e. g. for item generation) allows for an even better control of the items.

The sixth and final connection within the aspects of e-assessment is thus the connection between system design and data analysis. We have seen that this connection is less strong than the others. Data analysis may involve other data than just data from e-assessment and may thus happen outside an actual e-assessment system. Hence, designers of e-assessment systems do not commonly include components for that purpose. The only exception is the specific feature of adaptive e-assessment that relies quite much on continuous data analysis and thus is represented by appropriate components in the system design. Nevertheless, also the weak connection tells a lot about integration: To be able to perform a detailed analysis of assessment data outside the actual e-assessment system, appropriate data structures and interfaces must be built into the systems to export the required data in an appropriate format. In turn, system design can encapsulate analysis algorithms into well-defined components that can be used within an e-assessment system but also in other systems that can benefit from the same kind of data processing. Moreover, system design can help to tackle the legal issues that are connected to data processing via appropriate measures for authorization and pseudonymization.

22.2. Integration of E-Assessment into Context

We have also seen a connection between assessment processes and the context. Depending on the educational setting, assessments are more or less formal and thus involve more or less activities or even stakeholders. In very informal settings, complete phases of processes can be skipped, while formal settings may imply fixed decisions about e-assessment systems or item formats and thus limit the freedom in process design. In turn, the kernel of educational assessment treats organizers and authorities as rather abstract entities and thus makes deliberately no assumptions about the processes these stakeholders follow outside the actual assessment duties. Hence we can conclude that assessment processes must always be understood as integrated into a larger context with many additional processes. These processes may request and restrict certain features of the assessment process on the one hand, but on the other hand they require the assessment process to deliver the expected outcomes that are necessary to advance these context processes.

Another connection between e-assessment and its context is based on system design. We have seen that there is no strict border that allows to distinguish between e-assessment systems and other systems that incorporate e-assessment features. Moreover, not all e-assessment systems are the same, but include different components depending on the context in which they are used. At the same time, e-assessment system offer a variety of interfaces to access the system by human users for different purposes, as well as different technical interfaces and patterns on how to transfer data or integrate additional components. The need for integration into a context is thus a strong argument in favour of modular system design. We can thus conclude that technical integration of e-assessment

systems into context allows to provide interfaces and components for using them in other systems, while the context requests multiple ways to access existing e-assessment systems for different purposes and by different users.

A somewhat similar connection on a more conceptual level exists with respect to domain-specific data sources and tools. In that case, it is the context that provides a multitude of components, interfaces, standards and tools that may be beneficial or even necessary to be considered for the purpose of domain-specific e-assessment. Similarly, generic algorithms may be interested in using general knowledge facts from the context in order to process domain-specific assessment items. Domain-specific e-assessment is thus very much concerned with the conceptual integration of existing domain-specific features into e-assessment. In particular, we can conclude that the context provides standards and tools that prescribe decisions to be made during the design of domain-specific e-assessment features and algorithms. In turn, domain-specific data formats and tools strive to integrate as much knowledge facts from the context to make e-assessment more flexible and less dependant on hard coded rules and information that needs expensive frequent updates.

Finally, we have identified data analysis as an interface that connects plain assessment results with a context that cares about personal progress and improvements of educational quality. Despite that fact that advanced analysis techniques for competency measurement are also relevant for adaptive e-assessment, most analysis techniques have a much broader scope and goals far beyond the actual assessment. In particular, these goals may go beyond the questions asked by the educational assessment process itself, that were already mentioned in the previous section. Instead, goals like outcome prediction or some goals of item and answer analysis are concerned with the quality of educational processes in which assessment is only one of many events. Notably, these goals may even change the character of an assessment, if it becomes less important for assessing persons and more important for assessing the educational context. Thus we can conclude that data analysis is an interface between assessment and its context that provides additional value, while the context may provide additional data but also legal or ethical rules.

22.3. The Final Picture

All the different connections between the four aspects of e-assessment as well as the connections to the context can be summarized into one final picture as shown in figure 22.1. It details the idea that has been sketched in figure 1.1 in the introduction and thus turns the illustrative nature of that figure into an usable map.

We can now pick individual uni-directional or bi-directional connections from that map that we would like to explore in future research (see chapter 24 below) or that we would like to discuss in a teaching lesson. We can also use the map to find paths. For example, we can see that the context provides standards and tools to be used in domain-specific e-assessment in the first place. This connection propagates further, as domain-specific e-assessment prepares and offers those tools for integration into the general design of e-assessment systems. At the end of this way, system design in turn provides interfaces

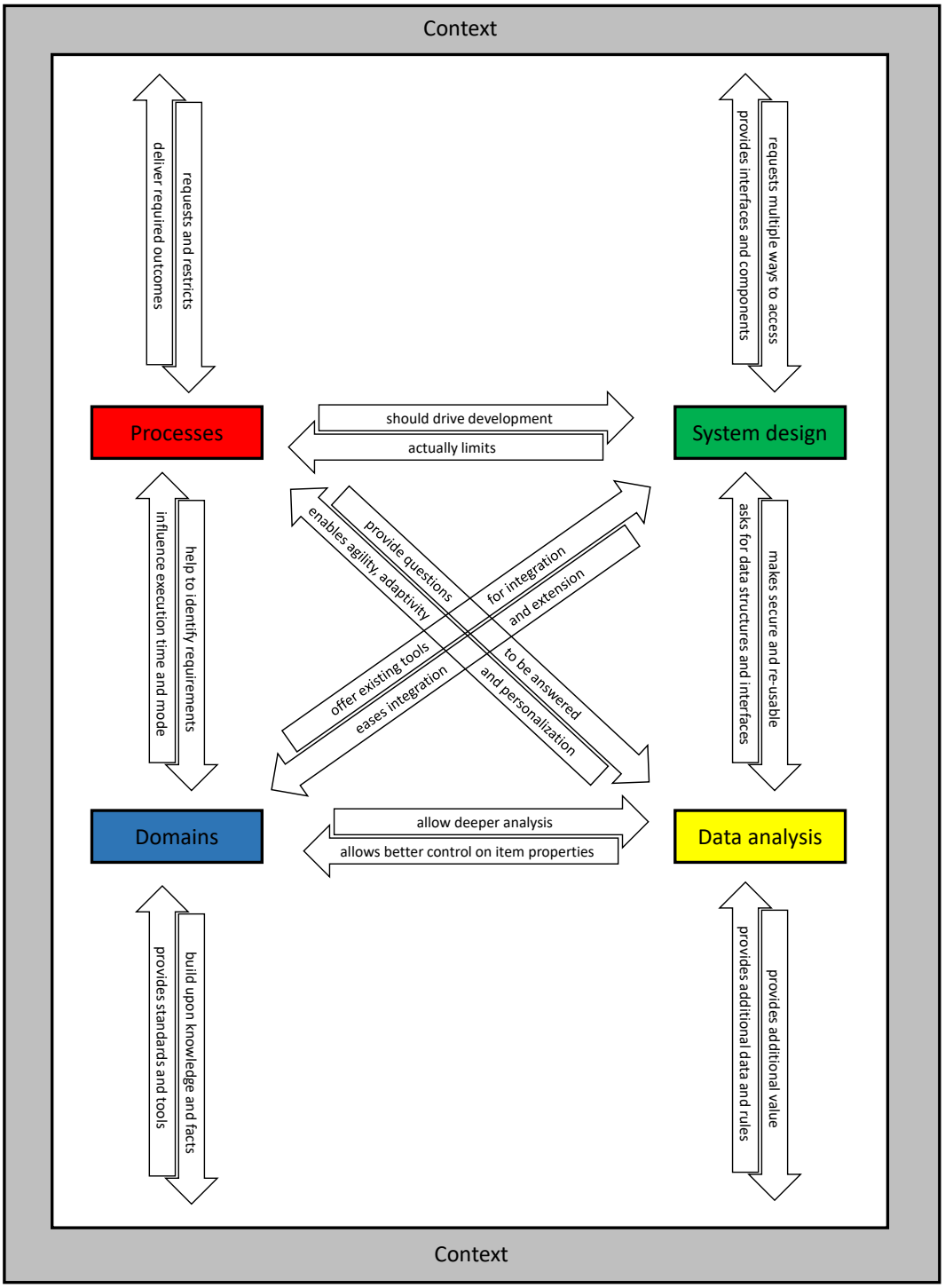


Figure 22.1.: The final picture of integrated e-assessment and its connections.

and components that cannot only be used in e-assessment systems, but also in broader context. That may also include variants or evolved versions of standards or tools that originally came from the context and tickled their way through e-assessment applications. As another example, we can also find circles in the map. We can start one of them with the observation that the existence or absence of domain-specific data-sources and tools influences the way in which processes can be executed. If a particular domain for instance cannot integrate techniques for fully automated item generation, we may see a remarkable amount of manual, time-consuming activities in an assessment process. That in turn is a driver for system design, since such activities call for new components that can help to reduce the manual effort. The circle is then closed if system design achieves to find integration mechanisms or extensions to existing components that allow to use new techniques for fully automated item generation that solve the problems of that particular domain.

The idea of using the final picture as a kind of map also stresses another aspect of integration: The different aspects of e-assessment are not only connected to each other when we plan and conduct an actual e-assessment, but they are also connected to each other when doing research to advance e-assessment. If we want to advance one of the aspects, we should thus always look at all of its connection to see whether either one of the connected aspects can support our endeavour or whether the progress in our research can also advance connected aspects.

23. Achievements

The aim of this publication was to define, explain and cover e-assessment as a subject of study that brings together research results from process modeling, software engineering, test item design and educational data analysis. Four distinct parts have been created that each tackle one of these four aspects. Each part summarizes its results and creates connections to the results from its preceding parts. Finally, chapter 22 draws the final picture of an integrated view on e-assessment. Based on that structure, this publications has created several achievements on two different levels: On a more abstract level, it has summarized the state-of-the-art in the four areas, and it has described and charted connections between these areas. On the more concrete level, it has created models, structures or catalogues as frameworks for orientation while solving actual challenges, and it has applied these in several case studies.

With respect to assessment processes, we have *created a notation for assessment process models and used it in some exemplary case studies.* The kernel of educational assessment that is part of the model made one explicit reference to e-assessment by providing an alpha dedicated specifically to e-assessment systems. Hence, we have seen that electronic assessments add some amount of complexity to the general concept of assessments by introducing new entities to interact with and new duties to be picked up. Moreover, we have seen that integrating these new elements into the existing collection of concepts, entities and duties related to assessments is possible without problems. Finally, we have seen that the sequencing of activities within the phase model raises important questions about the integration of tools into the educational process, since tools may restrict the freedom of choice in some didactic questions. Hence, we came to the conclusion that *not the process activities but the features offered by a particular e-assessment system are the key factors* that may ease or limit the integration in a particular process or institution.

These results directly relate to the achievements in the area of software engineering for e-assessment systems. Two *catalogues have been developed in that area, that contain components for e-learning and e-assessment systems and patterns on how to integrate these components* with each other, respectively. Components like “assessment generator” and “evaluator component” directly link to activities such as “author tests” and “create feedback”, so that we can use these results to relate system features and behaviour to states and activities in the assessment process. This helps to identify system components or features that must be included into a system that is supposed to support a particular type of assessment process. In turn, this also helps to identify missing interfaces or patterns for component integration. Since both catalogues are based on a large amount of actual system descriptions from literature and have additionally been validated in three case studies, they can also be used to *discuss the integration of e-assessment capabilities*

into other systems as well as the technical integration within an architecture based on a unified terminology.

Similar to the relation between process models and software components, there is also a direct relation between software components and domain-specific item handling. Components like "item generators" and "domain-specific expert systems" directly relate to the integration of domain-specific aspects into a general e-assessment system. Thus, it is another achievement to *identify several conceptual interfaces between the generic and domain-specific parts of an assessment item.* This in turn helps to design proper technical interfaces between system components as well as data storage within an e-assessment system that may respect the different structures of domain-specific data formats. Besides this technical contribution, we have also seen a *structured overview of several classes of techniques for input editors, data formats, item generation and answer evaluation.* That overview can be used to classify assessment items or assessment systems and compare them to each other based on their domain-specific capabilities or demands.

Finally, we have seen that the various types of data that are created in and processed by e-assessment systems form several interfaces via that e-assessment systems communicate with the surrounding context. Many goals of data processing relate to the context of an assessment, in which a single assessment is not an isolated event, but an event with consequences on the future development of persons, learning materials and assessment items. In order to illustrate the manifold ways in which different techniques can be combined, we used two case studies. These *demonstrate how advanced techniques can be used to gain valuable insight into learning behaviour from assessment data.* Process models for assessment thus do not only describe the process itself, but also talk about the context by naming the relevant data. We were able to *provide a rich tool set of different state-of-the-art techniques for data processing as well as an overview on the different goals and contexts they are used in.* These tools can help to reach various checkpoints for the states in the kernel for educational assessments and fill the related activity spaces with actual activities and techniques.

The insights into the various aspects of e-assessment allow to set up a teaching agenda for courses on e-assessment as part of a curriculum on educational technology. The process model from part I can serve both as an introduction and the guiding theme within that curriculum. It illuminates the various aspects of e-assessment and allows to get a broad overview without digging into the details too early. If there is no room for a whole course but only a single lecture on e-assessment, the process model can be used as well with additional explanation in the specific areas of interest of the students. Within a course, the overview can be followed by a series of individual lectures that each tackle one particular aspect of e-assessment by discussing the necessary software components from part II, their (potentially domain-specific) algorithms from part III and the data processed by them from part IV. Topics of such lectures can for example be "automated grading", "item generation" or "adaptive assessment", where each of these topics may contain enough material for more than one lecture. Such a course can be concluded by assembling all pieces together, thus talking about the way the individual software components can be combined to different types of e-assessment systems. Finally, the different options to use assessment results and data for other purposes beyond assessment

and the resulting ethical and legal issues can be discussed, which connects such a course to the broader field of educational technology.

24. Future Research Directions

Although this publication created some achievements, there are ample possibilities for future research. In fact, several achievements were about structured overviews, catalogues or similar fundamental work, that prepares the ground to start new endeavours. Some of those may concern only one of the four major aspects, while others will use one or more of the many connections we have found. The latter is not surprising, since the final picture of integrated e-assessment clearly illuminates the idea that none of the aspects can be advanced in isolation.

A vast space for further exploration and research is about the educational processes. So far, only a notation and a kernel of educational assessment has been created. Systematic studies based on literature as well as on interviews with practitioners, or observation of actual processes can help to gather much more details about activities and work products that occur when processes are enacted. The chosen notation allows to document these as assessment practices and patterns in a formalized but flexible manner. The resulting catalogue of assessment practices bears the potential to ease empirical studies on educational assessment due to a unified and common nomenclature for describing the actual scenarios. Moreover, practices and patterns can be used to explore the connections to software components, domain-specific tools and data analytics more systematically. The result of these research activities could be a rich map of assessment practices and patterns, where each possible set of meaningful assessment activities is associated with technical guidance on the one hand and empirical evidence on the other hand. Similar could be achieved for the broader scope of educational technology beyond e-assessment, by creating another kernel for teaching processes. Obviously, creating such a kernel is not an easy endeavour, at least if it should be able to cover various processes from single lessons up to complete courses. A reasonable starting point for that could be to create structural and hierarchical features that allow to formalize re-occurring assessments or complex assessments that are composed of several individual assessment events at different points in time. This approach can on the one hand build closely on top of the existing kernel and on the other hand help to structure long-running processes based on frequent occurrences of well-known assessment events.

In a similar way, software design for e-assessment systems can be explored much further than is has been done in this publication. In particular, the idea of design patterns can be explored further to come up with a more formalized catalogue similar to other examples of domain-specific software engineering. This will especially help to promote the technical aspects of integrated e-assessment, as different patterns for integrating e. g. domain-specific tools or data analysis algorithms can be explored. Similarly, different ways to integrated e-assessment components into other systems can also be covered by that research. In addition, more research on the design of individual e-assessment

systems can help to come up with more detailed descriptions of common components, their interfaces and possible variants. This could help to unify documentation and thus make it easier for system designers to choose appropriate components. It could even lead to the creation of additional standards that fill the gap between high-level standards like IMS-AF or (the withdrawn) IEEE LTSA on the one hand and low-level standards for very specific problems like IMS-LTI or IMS-QTI on the other hand. But even without new standards for the design of e-assessment systems, detailed descriptions of common interfaces for e-assessment components can help in other areas to come up with standardized ways to connect to those interfaces, e. g. for domain-specific e-assessment or advanced data analysis. The latter is particularly interesting, since we have mentioned earlier that system design can help to tackle legal issues in data analysis. There is already a lot of fundamental research work on these issues independent of the actual design of e-assessment systems, but little systematic research on how these features can actually be integrated in a usable way into existing e-assessment systems.

The aspect of domain-specific e-assessment is probably the one that offers the most opportunities for future research. We were only able to cover the most important aspects of that area on an abstract level and looked at three exemplary domains in the case studies. However, virtually any domain deserves a detailed mapping from the abstract concepts for input editors, data formats, item generation and answer processing to the actual needs and characteristics of that domain. We can expect to see challenging characteristics at least for some of these topics in most domains. At the same time, it is quite likely that individual solutions from one domain can be generalized to solve similar problems in another domain, even if that does not look very similar on the first glance. Most likely, a close collaboration between domain experts and assessment experts will be necessary to come up with item designs that properly reflect the important aspects of the domain, while item generation and answer evaluation can still be fully automated. At the same time, generic ways to encode domain-specific knowledge via approaches like ontologies need further investigations with regard to their use specifically in e-assessment. It would probably be a great success for domain-specific e-assessment if robust ways can be found to use arbitrary ontologies for item generation or answer evaluation. Similarly, there is space for more research on the flexible integration or creation of domain-specific input editors, that are a necessary prerequisite for assessment on higher competence levels in some domains.

The latter aspect directly leads to the aspect of data analysis as fourth field of research. In that area, we have seen a rich tool set of analysis techniques, that nevertheless appeared somewhat unconnected to each other. In particular, competence measurement is a quite old technique with a solid theory and known successful applications with fully implemented adaptive assessment systems, while other techniques are much younger and still experimental. Hence, there is a lot of space for more research on each of them and also on possible connections. Particularly, the connection to domain-specific assessments seems to be promising for future research. Domain-specific analysis techniques may allow to come up with detailed data about answer artifacts that allow to connect low-level technical properties to high-level concepts of competency measurement. Similarly, it is most likely that assessment items from different domains require different techniques in order to

improve item quality by item and answer analysis. It would thus be worthwhile to come up with domain-specific analysis algorithms that can be integrated into e-assessment systems and thus help item authors to monitor the quality of their assessments continuously in a detailed manner. At the same time, there seems to be little benefit in integrating each and every possible analysis technique into e-assessment systems. Instead, research towards standardized formats and interfaces for collecting and exchanging raw data from different systems can help to establish more flexible analysis solutions. That in turn stresses the need for answers to legal and ethical issues, if more data from different systems is integrated with each other.

Finally, the connections to the context of integrated e-assessment can be explored further. Integration means, that e-assessment is always just one piece of a larger picture: Conducting assessments is just one activity during educational processes; e-assessment systems are just one form of educational technology; domain-specific assessment items are just one characteristic of a scientific or educational domain; and assessment data is just the minor part of a large set of educational data. Hence, similar research as it has been conducted for this publication can be done in other areas of educational technology and the connections between these areas can be explored even further. It is most likely that some of the insights into integrated e-assessment are also true for other areas of educational technology and hence it would be interesting to find out whether one can finally come up with a more general picture of integrated educational technology.

25. Concluding Remarks

As we have seen in the previous chapters, this publication has created several achievements, but there are also many open questions that leave an open space for future research. In fact, this publication has neither asked a single, precise question, not has it answered one with scientific rigor. Nevertheless, this publication has created something useful: It started with the vague idea that e-assessment is something that is quite normal nowadays and thus integrated into teaching and learning in some way. Now there is a large map of various aspects of e-assessments and a lot of connections between each of them as well as to the context. That map may not always answer the question how strong a particular connection is or how it can be improved or why it is there at all. But it allows to see and name the connections and thus start to talk and research about them in a more structured way. Thus the mission of this publication is fulfilled: It defines, explains and covers e-assessment as a subject of study that brings together research results from process modeling, software engineering, test item design and educational data analysis.

Although the final picture is surely the ultimate result of that particular research, it must also be understood as a starting point. Since integrated e-assessment is now defined as a field of study, we can start to advance and evolve it. The earliest e-assessment systems on punch-cards do no longer exist today, and most likely today's e-assessment systems will no longer exist in 50 or 60 years. Teaching and learning may also see strong or even disruptive changes as it has recently with a sudden growth in online and distance learning – particularly due to a global pandemic, but also before. Since e-assessment is already integrated into today's practice, it has to advance and evolve as the context evolves. That may even imply that the final picture will need an update because new aspects come into play and new connections can be found. But that once again stresses the point that this publication is a starting point for research on integrated e-assessment rather than an ultimate result.

A. Tables

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
Accessibility	An Accessibility Component contains the data structures and interfaces responsible for describing the cognitive, technical and physical preferences for the learner, disability, eligibility and language capabilities. These describe the learner's capabilities to interact with the learning environment.	Student model
Activity	An Activity Component contains the data structures and interfaces responsible for describing the learning materials produced by the learner. This may consist of the education/training, work and service (military, community, voluntary, etc.) record and products (excluding formal awards). This information may include the descriptions of the courses undertaken and the records of the corresponding assessment.	(General data storage)
Affiliation	An Affiliation Component contains the data structures and interfaces responsible for describing the organization affiliations associated with the learner e. g., professional memberships.	(General data storage)
Assessment	An Assessment Component contains all of the necessary instructions to enable the presentation of the associated Items, variable sequencing of the Items, the aggregated scoring for all of the Sections/Items to produce the final score(s), and the corresponding feedback. The Section Component is used to construct the appropriate hierarchical Section/Item groups. The results from an Assessment can be reported using the Result Report Component.	Student frontend, Assessment generator, Evaluator component

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

(Table continued from previous page)

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
Competency	A Competency Component contains the data structures and interfaces responsible for describing the skills the learner has acquired. These skills may be associated with some formal or informal training or work history (described in the 'activity') and formal awards (described in the QCL Component).	Student model
Content	A Content Component contains the data structures and interfaces responsible for describing the logical structure, physical layout and associated presentation styles of the learning material. The material itself can take the form of text, image, video, audio, applet and an executable application. Alternative content issues are also addressed to support multi-lingual systems and to ensure the accessibility of the material.	(General data storage)
Course Catalogue	A Course Catalogue Component contains the data structures and interfaces responsible for listing the set of courses available to a learner. The catalogue contains at least the title, identifiers, and the start/end dates of the course. Other information may also be made available depending on the type of catalogue.	(General data storage)
Glossary	A Glossary Component contains the data structures and interfaces responsible for defining a glossary of key words and phrases. It is possible to define different types of glossary and to have hierarchical glossaries.	Domain knowledge model
Goal	A Goal Component contains the data structures and interfaces responsible for describing the learner's personal objectives and aspirations. These descriptions may also include information for monitoring the progress in achieving the goals. A goal can be defined in terms of subgoals.	Student model

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

(Table continued from previous page)

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
Grade-book	A Grade-book Component contains the data structures and interfaces responsible for recording the grades, comments, attendance, and scores for a student or group.	(General data storage)
Group	The Group Component contains the data structures and interfaces responsible for describing a set of related objects. Each member of a group will be unique. The type of relationship is implicit in the type group.	—
Interest	An Interest Component contains the data structures and interfaces responsible for describing the hobbies and other recreational activities of a learner. These interests may have formal awards (as described in the associated 'QCL Component'). Electronic versions of the products of these interests may also be contained.	—
Item	An Item Component contains all of the necessary instructions to enable the presentation of the associated question, the response processing to produce the set of scores, and the corresponding feedback. The results from an Item can be reported using the Result Report Component.	Student frontend, (General data storage)
LOM	A LOM Component contains the data structures and interfaces responsible for labelling an associated resource. The way in which the meta-data is associated with the resource is established via the appropriate component definition.	—
Manifest	A Manifest Component contains the data structures and interfaces responsible for constructing a content package. A content package is the IMS's generic aggregation and packaging mechanism and as such it can be used to package any type of content.	—

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

(Table continued from previous page)

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
Object-bank	The Object-bank Component is used to enable the grouping of Items and Sections in a data-bank. This data-bank is then used as the repository that is referenced by Assessments and Sections.	(General data storage)
Outcomes Processing	An Outcome Processing Component contains the data structures and interfaces that provide the mechanism through which the scores from any combination of Item Components and Section Components can be combined using one of the predefined algorithms. The set of algorithms available for the aggregation are accessed through the outcomes processing component operators.	Evaluator component
Party	The Party Component defines the data structures and interfaces responsible for describing an individual or an organization. The information includes names, addresses, demographics, agents, and contact information.	—
PLIRI	The PLIRI Component defines the data structures and interfaces responsible for defining and allocating globally unique and locally unique identifiers.	—
Profile	The Profile Component defines the data structures and interfaces responsible for constructing and manipulating a learner's profile. The profile may vary from a detailed life long learning log to a short summary of personal details. A learner may have more than one profile and each profile may be distributed across several profile servers.	Student model

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

(Table continued from previous page)

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
QCL	A QCL Component defines the data structures and interfaces responsible for describing the qualifications, certifications and licenses awarded to the learner i. e., the formally recognized products of their learning and work history. This includes information on the awarding body and may also include electronic copies of the actual documents.	—
Relationship	A Relationship Component defines the data structures and interfaces responsible defining the relations between other Components. All of the relationship information has been removed from the other components to enable these to be manipulated using a single independent component. The relationship is defined using a particular vocabulary and the components are identified using the appropriate PLIRI.	—
Result Report	The Result Report Component is used to report the results from any form of assessment e. g., a Test, Quiz, etc. The assessment may or may not be based upon the Assessment, Section or Item Components. Any level of detail can be reported from the assessment with the exception of tracking level information.	Data transfer component
Section	A Section Component contains all of the necessary instructions to enable the presentation of the associated Items, variable sequencing of the Sections/Items, the aggregated scoring for all of the Items to produce the Section score(s), and the corresponding feedback. The Section is used to construct hierarchical Section/Item groups The results from a Section can be reported using the Result Report Component.	Student frontend, Assessment generator, Evaluator component

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

(Table continued from previous page)

Component Name in IMS-AF	Component Description in IMS-AF	Mapping to section 6.1
SecurityKey	The SecurityKey Component defines the data structures and interfaces responsible for storing the passwords and security codes that are to be used when communicating with the learner.	(General data storage)
Sequencing	A Sequencing Component defines the data structures and interfaces responsible for describing the set of possible presentation sequences for the collection of content resources.	—
Syllabus	A Syllabus Component defines the data structures and interfaces responsible for representing the syllabus for a course.	—
Table of Contents	A Table of Contents Component defines the data structures and interfaces responsible for representing the table of contents of a list of related objects e. g., a content package.	—
Transcript	A Transcript Component contains the data structures and interfaces responsible for describing a learner's transcript i. e., the summary records of the academic performance at an institution. This information may contain an arbitrary level of detail and so there is no proscribed structure for a transcript. This component contains no layout information for the transcript.	(General data storage)
Vocabulary	A Vocabulary Component contains the data structures and interfaces for representing a vocabulary. This vocabulary may be constructed as a simple list or a complex taxonomy.	Domain knowledge model

Table A.1.: Mapping from components listed in the IMS-AF standard [129] to the terms used in section 6.1. Entry "—" indicates that there is no proper mapping. Entries in brackets indicate that the respective component has only been discussed implicitly in section 6.1.

Bibliography

[1] IEEE Standard for Learning Technology-Learning Technology Systems Architecture (LTSA), 2003.

[2] Essence - Kernel and Language for Software Engineering Methods, Dec 2015. URL `http://www.omg.org/spec/Essence/1.1`.

[3] A. Abelló, M. E. Rodríguez, T. Urpí, X. Burgués, M. J. Casany, C. Martín, and C. Quer. LEARN-SQL: Automatic Assessment of SQL Based on IMS QTI Specification. In *Eighth IEEE International Conference on Advanced Learning Technologies*, pages 592–593, July 2008. doi: 10.1109/ICALT.2008.27.

[4] Naveed Afzal and Ruslan Mitkov. Automatic generation of multiple choice questions using dependency-based semantic relations. *Soft Computing*, 18(7):1269–1281, Jul 2014. ISSN 1433-7479. doi: 10.1007/s00500-013-1141-4.

[5] M. Al-Smadi and C. Gütl. SOA-based architecture for a generic and flexible e-assessment system. In *IEEE EDUCON 2010 Conference*, pages 493–500, April 2010. doi: 10.1109/EDUCON.2010.5492537.

[6] M. Al-Yahya. OntoQue: A Question Generation Engine for Educational Assesment Based on Domain Ontologies. In *2011 IEEE 11th International Conference on Advanced Learning Technologies*, pages 393–395, July 2011. doi: 10.1109/ICALT.2011.124.

[7] Hanan Aldowah, Hosam Al-Samarraie, and Wan Mohamad Fauzy. Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telematics and informatics*, (37):13–49, 2019. ISSN 0736-5853. doi: 10.1016/j.tele.2019.01.007.

[8] Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. Ontology-Based Multiple Choice Question Generation. *KI - Künstliche Intelligenz*, 30(2):183–188, 2016. ISSN 1610-1987. doi: 10.1007/s13218-015-0405-9.

[9] Christine Alvarado and Randall Davis. SketchREAD: A Multi-Domain Sketch Recognition Engine. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. Association for Computing Machinery. doi: 10.1145/1281500.1281545.

[10] Salha M. Alzahrani, Naomie Salim, and Ajith Abraham. Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods. *IEEE Transactions*

on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(2): 133–149, 2011.

[11] Mario Amelung, Katrin Krieger, and Dietmar Rösner. E-Assessment as a Service. *IEEE Transactions on Learning Technologies*, 4:162–174, 2011. ISSN 1939-1382. doi: doi.ieeecomputersociety.org/10.1109/TLT.2010.24.

[12] A. Amigud, J. Arnedo-Moreno, T. Daradoumis, and A.-E. Guerrero-Roldan. Open proctor: An academic integrity tool for the open learning environment. In *Lecture Notes on Data Engineering and Communications Technologies*, volume 8, pages 262–273, 2018. doi: 10.1007/978-3-319-65636-6_23.

[13] Heidi M. Anderson, Guadalupe Anaya, Eleanora Bird, and Deborah L. Moore. A Review of Educational Assessment. *American Journal of Pharmaceutical Education*, 69(1), 2005. doi: 10.5688/aj690112.

[14] E. Araujo, D. Serey, and J. Figueiredo. Qualitative aspects of students' programs: Can we make them measurable? In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, Oct 2016. doi: 10.1109/FIE.2016.7757725.

[15] Martin E. Arendasy and Markus Sommer. Using automatic item generation to meet the increasing item demands of high-stakes educational and occupational assessment. *Learning and Individual Differences*, 22(1):112–117, 2012. ISSN 1041-6080. doi: 10.1016/j.lindif.2011.11.005.

[16] Martin E. Arendasy, Andreas Hergovich, and Markus Sommer. Investigating the 'g'-saturation of various stratum-two factors using automatic item generation. *Intelligence*, 36(6):574–583, 2008. ISSN 0160-2896. doi: 10.1016/j.intell.2007.11.005.

[17] Martin E. Arendasy, Markus Sommer, and Friedrich Mayr. Using Automatic Item Generation to Simultaneously Construct German and English Versions of a Word Fluency Test. *Journal of Cross-Cultural Psychology*, 43(3):464–479, 2012. doi: 10.1177/0022022110397360.

[18] Goce Armenski and Marjan Gusev. E-Testing Based on Service Oriented Architecture. In *Proceedings of the 10th CAA Conference*, 2006.

[19] Goce Armenski and Marjan Gusev. The Architecture of an "Ultimate" e-Assessment System. In *Association for Information and Communication Technologies ICT-ACT*, 2009.

[20] Alexander Askinadze. Anwendung der Regressions-SVM zur Vorhersage studentischer Leistungen. In *Proceedings of the 28th GI-Workshop Grundlagen von Datenbanken, Nörten Hardenberg, Germany, May 24-27, 2016*, pages 15–20, 2016.

[21] Yigal Attali. Automatic Item Generation Unleashed: An Evaluation of a Large-Scale Deployment of Item Models. In Carolyn Penstein Rosé, Roberto Martínez-Maldonado, H. Ulrich Hoppe, Rose Luckin, Manolis Mavrikis, Kaska Porayska-Pomsta, Bruce McLaren, and Benedict du Boulay, editors, *Artificial Intelligence*

in Education, pages 17–29, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93843-1.

[22] J.M. Azevedo, E.P. Oliveira, and P.D. Beites. Using learning analytics to evaluate the quality of multiple-choice questions: A perspective with classical test theory and item response theory. *International Journal of Information and Learning Technology*, 36(4):322–341, 2019. doi: 10.1108/IJILT-02-2019-0023.

[23] Richard Bacon. Assessing the Use of a New QTI Assessment Tool within Physics. In *Proceedings of the 7th CAA Conference*, 2003.

[24] Trudy W. Banta and Catherine A. Palomba. *Assessment Essentials*. John Wiley & Sons Inc, 2nd edition, 2015. ISBN 9781118903322. URL `http://www.ebook.de/de/product/22734250/trudy_w_banta_catherine_a_palomba_assessment_essentials.html`.

[25] David J. Bartholomew. *Analysis of multivariate social science data*. Chapman & Hall/CRC statistics in the social and behavioral sciences series. CRC Press, Boca Raton, 2nd edition, 2008. ISBN 978-1-58488-960-1.

[26] X. Baró, R. Muñoz Bernaus, D. Baneres, and A.E. Guerrero-Roldán. Biometric tools for learner identity in e-assessment. In *Lecture Notes on Data Engineering and Communications Technologies*, volume 34, pages 41–65, 2020. doi: 10.1007/978-3-030-29326-0_3.

[27] Xavier Baró, Roger Muñoz Bernaus, David Baneres, and Ana Elena Guerrero-Roldán. Biometric Tools for Learner Identity in e-Assessment. In *Engineering Data-Driven Adaptive Trust-based e-Assessment Systems*, number 34 in Lecture Notes on Data Engineering and Communications Technologies, pages 41–65. Springer, 2020. doi: 10.1007/978-3-030-29326-0_3.

[28] Firat Batmaz and Chris J. Hinde. A diagram drawing tool for semi-automatic assessment of conceptual database diagrams. In M. Danson, editor, *Proc. 10th Int. Computer Assisted Assessment Conf. (CAA), Loughborough, England*, pages 71–84. Professional Development, Loughborough University, 2006.

[29] Firat Batmaz, Roger Stone, and Chris Hinde. Personalised Feedback With Semi-Automatic Assessment Tool For Conceptual Database Model. *Innovation in Teaching and Learning in Information and Computer Sciences*, 9, 2010. doi: 10.11120/ital.2010.09010105.

[30] D. Bañeres, I. Noguera, M. Elena Rodríguez, and A. Guerrero-Roldán. Using an intelligent tutoring system with plagiarism detection to enhance e-assessment. In *Lecture Notes on Data Engineering and Communications Technologies*, volume 23, pages 363–372, 2019. doi: 10.1007/978-3-319-98557-2_33.

[31] Maria Berski. Klausurprognose mittels Learning Analytics auf Basis von E-Assessment-Daten. Bachelor's thesis, Universität Duisburg-Essen, 2018.

[32] Daniel Bildhauer and Jürgen Ebert. Querying Software Abstraction Graphs. In *Working Session on Query Technologies and Applications for Program Comprehension (QTAPC 2008), collocated with ICPC 2008*, 2008.

[33] L. Bin, L. Jun, Y. Jian-Min, and Z. Qiao-Ming. Automated Essay Scoring Using the KNN Algorithm. In *2008 International Conference on Computer Science and Software Engineering*, volume 1, pages 735–738, Dec 2008. doi: 10.1109/CSSE.2008.623.

[34] M. Birjali, A. Beni-Hssane, and M. Erritali. A novel adaptive e-learning model based on big data by using competence-based knowledge and social learner activities. *Applied Soft Computing Journal*, 69:14–32, 2018. doi: 10.1016/j.asoc.2018.04.030.

[35] A. Birnbaum. Some latent trait models and their use in inferring an examinee's ability. In *Statistical theories of mental test scores*, pages 395–479. Addision-Wesley, 1968.

[36] Bill Blyth and Aleksandra Labovic. Using Maple to implement eLearning integrated with computer aided assessment. *International Journal of Mathematical Education in Science and Technology*, 40(7):975–988, 2009. doi: 10.1080/00207390903226856.

[37] K. Boussaha, F. Mokhati, and C. Zakaria. Architecture of a specific platform for training practical works: Integration of learners' assessment component. *International Journal of Technology Enhanced Learning*, 7(3):195–220, 2015. doi: 10.1504/IJTEL.2015.072809.

[38] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984.

[39] Stina Bridgeman, Michael T. Goodrich, Stephen G. Kobourov, and Roberto Tamassia. PILOT: An Interactive Tool for Learning and Grading. In *Proceedings of the Thirty-first SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '00, pages 139–143, New York, NY, USA, 2000. ACM. doi: 10.1145/330908.331843.

[40] Peter Brusilovsky and Eva Millán. User models for adaptive hypermedia and adaptive educational systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The adaptive web*, pages 3–53. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-72078-2. URL `http://dl.acm.org/citation.cfm?id=1768197.1768199`.

[41] Peter Brusilovsky and Sergey Sosnovsky. Individualized Exercises for Self-assessment of Programming Knowledge: An Evaluation of QuizPACK. *J. Educ. Resour. Comput.*, 5(3), September 2005. ISSN 1531-4278. doi: 10.1145/1163405.1163411.

[42] S. Bull, B. Wasson, M. Kickmeier-Rust, M.D. Johnson, E. Moe, C. Hansen, G. Meissl-Egghart, and K. Hammermueller. Assessing english as a second language: From classroom data to a competence-based open learner model. In *Proceedings of the 20th International Conference on Computers in Education, ICCE 2012*, pages 618–622, 2012.

[43] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture*, volume Volume 1, A System of Patterns. Wiley, 1996. ISBN 978-0-471-95869-7.

[44] M. Delgado Calvo-Flores, E. Gibaja Galindo, M. C. Pegalajar Jiménez, and O. Pérez Pineiro. Predicting students' marks from Moodle logs using neural network models. *Current Developments in Technology-Assisted Education*, 1(2):586–590, 2006.

[45] D. Carneiro, P. Novais, D. Durães, J.M. Pego, and N. Sousa. Predicting completion time in high-stakes exams. *Future Generation Computer Systems*, 92:549–559, 2019. doi: 10.1016/j.future.2018.01.061.

[46] María José Casany, Marc Alier, Enric Mayol, Jordi Piguillem, Nikolas Galanis, Francisco J. García-Peñalvo, and Miguel Ángel Conde. Moodbile: A Framework to Integrate m-Learning Applications with the LMS. *Journal of Research and Practice in Information Technology*, 44(2):129–149, 2012.

[47] D. Challis. Committing to quality learning through adaptive online assessment. *Assessment & Evaluation in Higher Education*, 30(5):519–527, 2005.

[48] Lilia Cheniti-Belcadhi, Nicola Henze, and Rafik Braham. Implementation of a personalized assessment web service. In *Sixth International Conference on Advanced Learning Technologies (ICALT 2006)*, pages 586–590. IEEE, 2006.

[49] Martin Chodorow and Jill Burstein. Beyond essay length: Evaluating e-rater®'s performance on TOEFL® essays. *ETS Research Report Series*, 2004(1):i–38, 2004. doi: 10.1002/j.2333-8504.2004.tb01931.x.

[50] Hervé Cholez, Nicolaus Mayer, and Thibaud Latour. Information Security Risk Management in Computer-Assisted Assessment Systems: First Step in Addressing Contextual Diversity. In *Proceedings of the 13th Computer-Assisted Assessment Conference (CAA 2010)*, 2010.

[51] Wallace Clark, Walter Nicholas Polakov, and Frank W Trabold. *The Gantt Chart: A working tool of management*. Ronald Press Company, 1922.

[52] Catherine Cleophas, Christoph Hoennige, Frank Meisel, and Philipp Meyer. Who's Cheating? Mining Patterns of Collusion from Text and Events in Online Exams. Available online at SSRN, 2021. URL `https://ssrn.com/abstract=3824821`.

[53] Arjeh M. Cohen, Hans Cuypers, Dorina Jibetean, Mark Spanbroek, O. Caprotti, E. Eixarch, D. Marques, and A. Pau. LeActiveMath Integrated CAS with OpenMath, June 2005.

[54] Sébastien Combéfis and Guillaume de Moffarts. Automated Generation of Computer Graded Unit Testing-Based Programming Assessments for Education. In *6th International Conference on Computer Science, Engineering and Information Technology (CSEIT 2019)*, 2019. doi: 10.5121/csit.2019.91308.

[55] John W. Connolly. Automated homework grading for large general chemistry classes. *Journal of Chemical Education*, 49(4):262, 1972.

[56] Alberto Corbí and Daniel Burgos Solans. Semi-Automated Correction Tools for Mathematics-Based Exercises in MOOC Environments. *International Journal of Artificial Intelligence and Interactive Multimedia*, 3(3):89–95, 2015.

[57] Georgina Cosma and Mike Joy. Towards a Definition of Source-Code Plagiarism. *IEEE Transactions on Education*, 51(2):195–200, 2008.

[58] Evandro Costa, Priscylla Silva, Marlos Silva, Emanuele Silva, and Anderson Santos. A Multiagent-Based ITS Using Multiple Viewpoints for Propositional Logic. In *Intelligent Tutoring Systems*, pages 640–641. Springer, 2012.

[59] Bronwen Cowie and Beverley Bell. A Model of Formative Assessment in Science Education. *Assessment in Education*, 6(1):101–116, 1999.

[60] L. J. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

[61] Declan Dagger, Alexander O'Connor, Séamus Lawless, Eddie Walsh, and Vincent P. Wade. Service-Oriented E-Learning Platforms: From Monolithic Systems to Flexible Services. *IEEE Internet Computing*, 11(3):28–35, May 2007. ISSN 1089-7801. doi: 10.1109/MIC.2007.70.

[62] Myles Danson, Bryan Dawson, and Tim Baseley. Large Scale Implementation of Question Mark Perception (V2.5) – Experiences at Loughborough University. In *Proceedings of the 5th Computer-Assisted Assessment Conference (CAA)*, 2001.

[63] Will M Davies, Yvonne Howard, Hugh C Davis, David E. Millard, and Niall Sclater. Aggregating Assessment Tools in a Service Oriented Architecture. In *9th International CAA Conference*, 2005.

[64] Evandro de Barros Costa, Angelo Perkusich, and Edilson Ferneda. From a Tridimensional View of Domain Knowledge to Multi-agent Tutoring System. In *Advances in Artificial Intelligence*, pages 61–72. Springer Berlin Heidelberg, 1998. doi: 10.1007/10692710_7.

[65] F. S. de Oliveira and S. Santos. PBLMaestro: A virtual learning environment for the implementation of problem-based learning approach in Computer education. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, Oct 2016. doi: 10.1109/FIE.2016.7757388.

[66] Paul Deane and Kathleen Sheehan. Automatic Item Generation via Frame Semantics: Natural Language Generation of Math Word Problems. Paper presented at the Annual Meeting of the National Council on Measurement in Education (Chicago, IL, April 22-24, 2003), April 2003. URL `https://eric.ed.gov/?id=ED480135`.

[67] John V. Dempsey, Brenda C. Litchfield, and Marcy P. Driscoll. Feedback, Retention, Discrimination Error, and Feedback Study Time. *Journal of Research on Computing in Education*, 25(3):303–326, 1993. doi: 10.1080/08886504.1993.10782053.

[68] John Dermo. e-Assessment and the student learning experience: A survey of student perceptions of e-assessment. *British Journal of Educational Technology*, 40 (2):203–214, 2009. doi: 10.1111/j.1467-8535.2008.00915.x.

[69] K. Derr, R. Hübl, and M.Z. Ahmed. Using test data for successive refinement of an online pre-course in mathematics. In *Proceedings of the European Conference on e-Learning, ECEL*, pages 173–180, 2015.

[70] Katja Derr, Reinhold Hübl, and Mohammed Zaki Ahmed. Using test data for successive refinement of an online pre-course in mathematics. In *14th European Conference on e-Learning ECEL*, pages 173–180, 2015.

[71] Michel C. Desmarais and Ryan S. Baker. A Review of Recent Advances in Learner and Skill Modeling in Intelligent Learning Environments. *User Modeling and User-Adapted Interaction*, 22(1-2):9–38, April 2012. ISSN 0924-1868. doi: 10.1007/s11257-011-9106-8.

[72] Vladan Devedzic and Andreas Harrer. Architectural Patterns in Pedagogical Agents. In *Intelligent Tutoring Systems (ITS 2002)*, volume 2363 of *Lecture Notes in Computer Science*, pages 81–90, 2002. doi: 10.1007/3-540-47987-2_13.

[73] Vladan Devedzic, Danijela Radovic, and Ljubomir Jerinic. On the Notion of Components for Intelligent Tutoring Systems. In *Intelligent Tutoring Systems (ITS 1998)*, volume 1452 of *Lecture Notes in Computer Science*, pages 504–513, 1998.

[74] Walter Dick, Lou Carey, and James O. Carey. *The Systematic Design of Instruction*. Pearson Education, 8th edition, 2014.

[75] Bogdan Drăgulescu, Marian Bucos, and Radu Vasiu. Predicting assignment submissions in a multi-class classification problem. *TEM Journal*, 4(3):244, 2015.

[76] Cynthia Dwork. Differential Privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-35907-9, 978-3-540-35907-4. doi: 10.1007/11787006_1.

[77] Jürgen Ebert and Angelika Franzke. A Declarative Approach to Graph Based Modeling. In G. Tinhofer E. Mayr, G. Schmidt, editor, *Graphtheoretic Concepts in Computer Science*, number 903 in LNCS, pages 38–50, Berlin, 1995. Springer Verlag.

[78] ECMA TC33 Task Group and ISEE Working Group. Reference Model for Frameworks of Software Engineering Environments. Technical Report ECMA TR/55, European Computer Manufacturers Association, June 1993. URL `https://www.ecma-international.org/publications/techreports/E-TR-055.htm`.

[79] edX Inc. Documentation, Chapter 10.14. External Grader, 2017. URL http://edx.readthedocs.io/projects/edx-partner-course-staff/en/latest/exercises_tools/external_graders.html. last accessed 2017-12-15.

[80] Eman El-Sheikh and Jon Sticklen. Generating Intelligent Tutoring Systems from Reusable Components and Knowledge-Based Systems. In *Intelligent Tutoring Systems: 6th International Conference, ITS 2002, Biarritz, France and San Sebastian, Spain, June 2-7, 2002. Proceedings*, pages 199–207. Springer, 2002.

[81] Cath Ellis. Broadening the scope and increasing the usefulness of learning analytics: The case for assessment analytics. *British Journal of Educational Technology*, 44 (4):662–664, July 2013. doi: 10.1111/bjet.12028.

[82] Susan Embretson and Xiangdong Yang. Automatic Item Generation and Cognitive Psychology. In C. R. Rao and S. Sinharay, editors, *Psychometrics*, volume 26 of *Handbook of Statistics*, pages 747–768. Elsevier, 2006. doi: 10.1016/S0169-7161(06)26023-1. URL http://www.sciencedirect.com/science/article/pii/S0169716106260231.

[83] T. Dary Erwin. *Assessing Student Learning and Development: A Guide to the Principles, Goals, and Methods of Determining College Outcomes.* Jossey-Bass, 1991.

[84] Nickolas Falkner, Rebecca Vivian, David Piper, and Katrina Falkner. Increasing the Effectiveness of Automated Assessment by Increasing Marking Granularity and Feedback Units. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 9–14, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2605-6. doi: 10.1145/2538862.2538896.

[85] B.E. Florián G, S.M. Baldiris, and R. Fabregat. A new competency-based e-assessment data model: Implementing the aeea proposal. In *IEEE Education Engineering Conference, EDUCON 2010*, pages 473–480, 2010. doi: 10.1109/EDUCON.2010.5492538.

[86] Peter W. Foltz, Darrell Laham, and Thomas K. Landauer. The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1(2):939–944, 1999.

[87] Daniela Fonte, Ismael Vilas Boas, Daniela da Cruz, Alda Lopes Gançarski, and Pedro Rangel Henriques. Program analysis and evaluation using QUIMERA. In *ICEIS'2012-14th International Conference on Enterprise Information Systems*, volume 2, pages 209–219, 2012.

[88] Muriel Foulonneau. Generating Educational Assessment Items from Linked Open Data: The Case of DBpedia. In Raúl García-Castro, Dieter Fensel, and Grigoris Antoniou, editors, *The Semantic Web: ESWC 2011 Workshops*, pages 16–27. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-25953-1.

[89] Muriel Foulonneau and Eric Ras. Using Educational Domain Models for Automatic Item Generation Beyond Factual Knowledge Assessment. In Davinia Hernández-Leo, Tobias Ley, Ralf Klamma, and Andreas Harrer, editors, *Proceedings of EC-TEL 2013: Scaling up Learning for Sustained Impact*, pages 442–447, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40814-4.

[90] A. Frey and N. N. Seitz. Multidimensional Adaptive Testing in Educational and Psychological Measurement: Current state and future challenges. *Studies in Educational Evaluation*, 35:89–94, 2009.

[91] J. Gamulin, O. Gamulin, and D. Kermek. The application of formative e-assessment data in final exam results modeling using neural networks. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 726–730, 2015.

[92] J. Gamulin, O. Gamulin, and D. Kermek. The application of formative e-assessment data in final exam results modeling using neural networks. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings*, pages 726–730, 2015. doi: 10.1109/MIPRO.2015.7160367.

[93] Niels Gandraß and Axel Schmolitzky. Automatisierte Bewertung von Java-Programmieraufgaben im Rahmen einer Moodle E-Learning Plattform. In *Proceedings of the Fourth Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2019)*, pages 3–10, 2019. doi: 10.18420/abp2019-1.

[94] Henry Laurence Gantt. *Work, Wages, and Profits*. Engineering Magazine Company, 2 edition, 1913.

[95] Alicia García-Holgado and Francisco José García-Peñalvo. Architectural pattern to improve the definition and implementation of eLearning ecosystems. *Science of Computer Programming*, 129:20 – 34, 2016. ISSN 0167-6423. doi: 10.1016/j.scico.2016.03.010. URL `http://www.sciencedirect.com/science/article/pii/S0167642316300259`. Special issue on eLearning Software Architectures.

[96] Robert Garmann. Ein Format für Bewertungsvorschriften in automatisiert bewertbaren Programmieraufgaben. In Niels Pinkwart and Johannes Konert, editors, *DeLFI 2019 - Die 17. Fachtagung Bildungstechnologien, 16.-19. September 2019, Berlin*, volume P-297 of *LNI*, pages 103–114. Gesellschaft für Informatik e.V., 2019. doi: 10.18420/delfi2019_73.

[97] Robert Garmann, Felix Heine, and Peter Werner. Grappa - die Spinne im Netz der Autobewerter und Lernmanagementsysteme. In *DeLFI 2015 - Die 13. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI), München, 1.-4. September 2015*, pages 169–181, 2015. URL `http://subs.emis.de/LNI/Proceedings/Proceedings247/article20.html`.

[98] David Gañán, Santi Caballé, Robert Clarisó, Jordi Conesa, and David Bañeres. ICT-FLAG: a web-based e-assessment platform featuring learning analytics and gamification. *International Journal of Web Information Systems*, 13(1):25–54, 2017.

[99] V. Geetha, D. Chandrakala, R. Nadarajan, and C.K. Dass. A bayesian classification approach for handling uncertainty in adaptive e-assessment. *International Review on Computers and Software*, 8(4):1045–1052, 2013.

[100] Mark J. Gierl and Thomas M. Haladyna, editors. *Automatic Item Generation*. Routledge, 2013. ISBN 9780415897501.

[101] Mark J. Gierl and Hollis Lai. The Role of Item Models in Automatic Item Generation. *International Journal of Testing*, 12(3):273–298, 2012. doi: 10.1080/15305058.2011.635830.

[102] Mark J. Gierl and Hollis Lai. Using Automatic Item Generation to Create Solutions and Rationales for Computerized Formative Testing. *Applied Psychological Measurement*, 42(1):42–57, 2018. doi: 10.1177/0146621617726788.

[103] Mark J. Gierl, Jiawen Zhou, and Cecilia Alves. Developing a Taxonomy of Item Model Types to Promote Assessment Engineering. *The Journal of Technology, Learning and Assessment*, 7(2), Dec. 2008. URL `https://ejournals.bc.edu/index.php/jtla/article/view/1629`.

[104] Mark J. Gierl, Hollis Lai, and Simon R. Turner. Using automatic item generation to create multiple-choice test items. *Medical Education*, 46(8):757–765, 2012. doi: 10.1111/j.1365-2923.2012.04289.x.

[105] George Goguadze. Representation for Interactive Exercises. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *Intelligent Computer Mathematics*, pages 294–309, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-02614-0.

[106] George Goguadze and Erica Melis. Combining Evaluative and Generative Diagnosis in ACTIVEMATH. In *AIED*, pages 668–670, 2009.

[107] Giorgi Goguadze. *Active Math - Generation and Reuse of Interactive Exercises using Domain Reasoners and Automated Tutorial Strategies*. PhD thesis, Universität des Saarlandes, 5 2011.

[108] Javier Gonzalez-Sanchez, Maria Elena Chavez-Echeagaray, Kurt VanLehn, Winslow Burleson, Sylvie Girard, Yoalli Hidalgo-Pontet, and Lishan Zhang. A System Architecture for Affective Meta Intelligent Tutoring Systems. In *International Conference on Intelligent Tutoring Systems*, pages 529–534. Springer, 2014.

[109] Jorge Fontenla González, Manuel Caeiro Rodríguez, Martín Llamas Nistal, Elio Sancristobal, and Manuel Castro. A middleware for the integration of third-party learning tools in SOA-based Learning Management Systems: Supporting instance

management and data transfer. In *IEEE EDUCON 2010 Conference*, pages 869–877, April 2010. doi: 10.1109/EDUCON.2010.5492487.

[110] Martin Greenhow and Mundeep Gill. Setting objective tests in mathematics using QM Perception. In *Proceedings of the 8th CAA Conference*, 2004.

[111] Sebastian Gross, Bassam Mokbel, Barbara Hammer, and Niels Pinkwart. Feedback Provision Strategies in Intelligent Tutoring Systems Based on Clustered Solution Spaces. In Jörg Desel, Joerg M. Haake, and Christian Spannagel, editors, *DeLFI 2012: Die 10. e-Learning Fachtagung Informatik*, pages 27–38, Hagen, Germany, 2012. ISBN 978-3885796015.

[112] Wolfgang Gräther, Sabine Kolvenbach, Rudolf Ruland, Julian Schütte, Christof Torres, and Florian Wendland. Blockchain for education: lifelong learning passport. In *Proceedings of 1st ERCIM Blockchain Workshop 2018*. European Society for Socially Embedded Technologies (EUSSET), 2018.

[113] Bettina Grün and Achim Zeileis. Automatic Generation of Exams in R. *Journal of Statistical Software*, 29(1):1–14, 2009. ISSN 1548-7660. doi: 10.18637/jss.v029.i10. URL `https://www.jstatsoft.org/index.php/jss/article/view/v029i10`.

[114] Marjan Gusev, Sasko Ristov, and Goce Armenski. Interactive adaptivity in Assessment as a Service. In *2013 International Conference on Interactive Collaborative Learning (ICL)*, pages 588–595, Sept 2013. doi: 10.1109/ICL.2013.6644660.

[115] Marjan Gusev, Sasko Ristov, Goce Armenski, Goran Velkoski, and Krste Bozinoski. E-Assessment Cloud Solution: Architecture, Organization and Cost Model. *iJET*, 8(Special Issue 2):55–64, 2013.

[116] H5P.org. H5P Documentation. URL `https://h5p.org/documentation`. Last accessed: 2017-12-01.

[117] Fahima Hajjej, Yousra Bendaly Hlaoui, and Leila Jemni Ben Ayed. A Generic E-Assessment Process Development Based on Reverse Engineering and Cloud Services. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 157–165, April 2016. doi: 10.1109/CSEET.2016.49.

[118] Andreas Harrer and Alke Martens. Towards a Pattern Language for Intelligent Teaching and Training Systems. In *Intelligent Tutoring Systems (ITS 2006)*, volume 4053 of *Lecture Notes in Computer Science*, pages 298–307, 2006.

[119] Andreas Harrer, Niels Pinkwart, Bruce M. McLaren, and Oliver Scheuer. The Scalable Adapter Design Pattern: Enabling Interoperability Between Educational Software Tools. *TLT*, 1(2):131–143, 2008. doi: 10.1109/TLT.2008.18.

[120] Ian Harwood and Bill Warburton. Thinking the Unthinkable: Using Project Risk Management when Introducing Computer-assisted Assessments. In *Proceedings of the 8th Computer Assisted Assesstment Conference (CAA 2004)*, 2004.

[121] Ioannis Hatzilygeroudis, Constantinos Koutsojannis, Constantinos Papavlasopoulos, and Jim Prentzas. Knowledge-based adaptive assessment in a Web-based intelligent educational system. In *Sixth International Conference on Advanced Learning Technologies (ICALT 2006).*, pages 651–655. IEEE, 2006.

[122] Tom-Michael Hesse, Axel Wagner, and Barbara Paech. Automated assessment of C++ programming exercises with unit tests. In *Proceedings of the First Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2013), Hannover, Germany, October 28, 2013.*, 2013. URL `http://ceur-ws.org/Vol-1067/abp2013_submission_7.pdf`.

[123] Heinz Holling, Jonas P. Bertling, and Nina Zeuch. Automatic item generation of probability word problems. *Studies in Educational Evaluation*, 35(2):71–76, 2009. ISSN 0191-491X. doi: 10.1016/j.stueduc.2009.10.004. Assessment of Competencies.

[124] Jack Hollingsworth. Automatic graders for programming classes. *Communications of the ACM*, 3(10):528–529, October 1960. ISSN 0001-0782. doi: 10.1145/367415.367422.

[125] Edmond Holohan, Mark Melia, Declan McMullen, and Claus Pahl. The Generation of E-Learning Exercise Problems from Subject Ontologies. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 967–969, July 2006. doi: 10.1109/ICALT.2006.1652605.

[126] Peter Hubwieser, Marc Berges, Michael Striewe, and Michael Goedicke. Towards Competency Based Testing and Feedback. In *Proceedings of IEEE Global Engineering Education Conference (EDUCON)*, pages 517–526, 2017. doi: 10.1109/EDUCON.2017.7942896.

[127] Lukas Iffländer, Alexander Dallmann, Philip Daniel-Beck, and Marianus Ifland. PABS - a Programming Assignment Feedback System. In *Proceedings of the Second Workshop "Automatische Bewertung von Programmieraufgaben"*, 2015.

[128] Marko Ikonen, Elena Pirinen, Fabian Fagerholm, Petri Kettunen, and Pekka Abrahamsson. On the impact of Kanban on software project work: An empirical case study investigation. In *Proceedings of the 16th IEEE International Conference on Engineering of Complex Computer Systems*, ICECCS, pages 305–314. IEEE, 2011. doi: 10.1109/ICECCS.2011.37.

[129] IMS-AF. IMS Abstract Framework: Applications, Services, and Components, 2003. URL `http://www.imsglobal.org/af/afv1p0/imsafascv1p0.html`.

[130] IMS-LTI. IMS Learning Tools Integration Specification, 2012. URL `https://www.imsglobal.org/specs/ltiv1p1p1`.

[131] IMS-QTI. IMS Question & Test Interoperability Specification, 2016. URL `http://www.imsglobal.org/question/`.

[132] N. Iwane, C. Gao, and M. Yoshida. Question Generation for Learner Centered Learning. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pages 330–332, July 2013. doi: 10.1109/ICALT.2013.102.

[133] Ambikesh Jayal and Martin Shepperd. The Problem of Labels in E-Assessment of Diagrams. *J. Educ. Resour. Comput.*, 8(4):12:1–12:13, January 2009. ISSN 1531-4278. doi: 10.1145/1482348.1482351.

[134] Christoph Jobst. Potenziale neuer Fragetypen für die Naturwissenschaften. In *Grundfragen Multimedialen Lehrens und Lernens (GML² 2015)*, pages 145–152, 2015.

[135] David H. Johnson and Roger T. Johnson. *Meaningful Assessment: A Manageable and Cooperative Process*. Pearson, 2002. ISBN 978-0205327621.

[136] Jelena Jovanović, Dragan Gašević, Christopher Brooks, Vladan Devedžić, and Marek Hatala. LOCO-analyst: A tool for raising teachers' awareness in online learning environments. In *European Conference on Technology Enhanced Learning*, pages 112–126. Springer, 2007.

[137] Violet Kafa, Marcellus Siegburg, and Janis Voigtlander. Exercise Task Generation for UML Class/Object Diagrams, via Alloy Model Instance Finding. In *SACLA 2019*, number 1136 in CCIS, 2019. doi: 10.1007/978-3-030-35629-3_8.

[138] Mustafa Kaiiali, Armagan Ozkaya, Halis Altun, Hatem Haddad, and Marc Alier. Designing a Secure Exam Management System (SEMS) for M-Learning Environments. *TLT*, 9(3):258–271, 2016. doi: 10.1109/TLT.2016.2524570.

[139] E. Kashy, B. M. Sherrill, Y. Tsai, D. Thaler, D. Weinshank, M. Engelmann, and D. J. Morrissey. CAPA-An integrated computer-assisted personalized assignment system. *American Journal of Physics*, 61:1124–1130, December 1993. doi: 10.1119/1.17307.

[140] Clauvice Kenfack, Roger Nkambou, Serge Robert, Ange Adrienne Nyamen Tato, Janie Brisson, and Pamela Kissok. A Brief Overview of Logic-Muse, an Intelligent Tutoring System for Logical Reasoning Skills. In *Intelligent Tutoring Systems (ITS 2016)*, 2016.

[141] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)*, 19(1):1–43, 2018. doi: 10.1145/3231711.

[142] Mohammad Khalil and Martin Ebner. Learning Analytics: Principles and Constraints. In *EdMedia+ Innovate Learning*, pages 1789–1799. Association for the Advancement of Computing in Education (AACE), 2015.

[143] Christophe Kiennert, Malinka Ivanova, Anna Rozeva, and Joaquin Garcia-Alfaro. Security and Privacy in the TeSLA Architecture. In *Engineering Data-Driven Adaptive Trust-based e-Assessment Systems*, number 34 in Lecture Notes on Data

Engineering and Communications Technologies, pages 85–108. Springer, 2020. doi: 10.1007/978-3-030-29326-0_5.

[144] Hidemasa Kimura, Jumpei Hayashi, Yuichi Demise, Dai Hasegawa, and Hiroshi Sakuta. The effects of listening agent in speech-based on-line test system. In *IEEE Global Engineering Education Conference, EDUCON 2015, Tallinn, Estonia, March 18-20, 2015*, pages 366–370, 2015. doi: 10.1109/EDUCON.2015.7095998.

[145] Alexander Kiy, Volker Wölfert, and Ulrike Lucke. Technische Unterstützung zur Durchführung von Massenklausuren. In *Die 14. E-Learning Fachtagung Informatik (DeLFI 2016)*, 2016.

[146] Eckhard Klieme, Hermann Avenarius, Werner Blum, Peter Döbrich, Hans Gruber, Manfred Prenzel, Kristina Reiss, Kurt Riquarts, Jügen Rost, Heinz-Elmar Tenorth, and Helmut Johannes Vollmer. *The Development of National Educational Standards. An Expertise.* BMBF: education reform. Bundesministerium für Bildung und Forschung, Bonn, 2004.

[147] Manon Knockaert and Nathan De Vos. Ethical, Legal and Privacy Considerations for Adaptive Systems. In *Engineering Data-Driven Adaptive Trust-based e-Assessment Systems*, number 34 in Lecture Notes on Data Engineering and Communications Technologies, pages 267–296. Springer, 2020. doi: 10.1007/978-3-030-29326-0_12.

[148] Ari Korhonen and Lauri Malmi. Algorithm Simulation with Automatic Assessment. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSEconference on Innovation and Technology in Computer Science Education*, ITiCSE '00, pages 160–163, New York, NY, USA, 2000. ACM. ISBN 1-58113-207-7. doi: 10.1145/343048.343157.

[149] Aravind K. Krishna and Amruth N. Kumar. A Problem Generator to Learn Expression: Evaluation in CSI, and Its Effectiveness. In *Proceedings of the Sixth Annual CCSC Northeastern Conference on The Journal of Computing in Small Colleges*, CCSC '01, pages 34–43, USA, 2001. Consortium for Computing Sciences in Colleges. URL `http://dl.acm.org/citation.cfm?id=378593.378659`.

[150] Johannes Krugel, Peter Hubwieser, Michael Goedicke, Michael Striewe, Mike Talbot, Christoph Olbricht, Melanie Schypula, and Simon Zettler. Automated Measurement of Competencies and Generation of Feedback in Object-Oriented Programming Courses. In *2020 IEEE Global Engineering Education Conference, EDUCON 2020, Porto, Portugal, April 27-30, 2020*, pages 329–338, 2020. doi: 10.1109/EDUCON45650.2020.9125323.

[151] Klaus D. Kubinger. Adaptive testing. In Karl Schweizer and Christine DiStefano, editors, *Principles and methods of test construction.* Hogrefe Publishing Boston, MA, 2016.

[152] Klaus D. Kubinger, Jan Steinfeld, Manuel Reif, and Takuya Yanagida. Biased (conditional) parameter estimation of a Rasch model calibrated item pool adminis-

tered according to a branched testing design. *Psychological Test and Assessment Modeling*, 54(4):450–460, 2012.

[153] Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. A Systematic Review of Automatic Question Generation for Educational Purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204, 2020. doi: 10.1007/s40593-019-00186-y.

[154] Filiz Kurt-Karaoglu, Nils Schwinning, Michael Striewe, Björn Zurmaar, and Michael Goedicke. A Framework for Generic Exercises with Mathematical Content. In *Proceedings of the International Conference on Learning and Teaching in Computing and Engineering (LaTiCE 2015)*, pages 70–75, 2015.

[155] Maya Kurup, Jim E. Greer, and Gordon I. McCalla. The Fawlty Article Tutor. In *Intelligent Tutoring Systems (ITS 1992)*, 1992.

[156] Bastian Küppers, Marius Politze, and Ulrik Schroeder. Reliable e-Assessment with GIT - Practical Considerations and Implementation. In *EUNIS 23rd Annual Congress*, 2017.

[157] Hollis Lai, Mark J. Gierl, Claire Touchie, Debra Pugh, André-Philippe Boulais, and André De Champlain. Using Automatic Item Generation to Improve the Quality of MCQ Distractors. *Teaching and Learning in Medicine*, 28(2):166–173, 2016. doi: 10.1080/10401334.2016.1146608. PMID: 26849247.

[158] Dugan Laird, Elwood F. Holton, and Sharon S. Naquin. *Approaches to Training and Development*. BASIC BOOKS, 2003. ISBN 978-0738206981. URL `http://www.ebook.de/de/product/4242753/dugan_laird_elwood_f_holton_sharon_s_naquin_approaches_to_training_and_development.html`.

[159] Maryline Laurent and Samia Bouzefrane, editors. *Digital identity management*. ISTE, London, 2015.

[160] Zsolt Lavicza. Examining the use of Computer Algebra Systems in university-level mathematics teaching. *Journal of Computers in Mathematics and Science Teaching*, 28(2):99–111, April 2009. ISSN 0731-9258. URL `http://www.editlib.org/p/30303`.

[161] Nguyen-Thinh Le, Tomoko Kojiri, and Niels Pinkwart. Automatic Question Generation for Educational Applications – The State of Art. In Tien van Do, Hoai An Le Thi, and Ngoc Thanh Nguyen, editors, *Advanced Computational Methods for Knowledge Engineering*, pages 325–338, Cham, 2014. Springer International Publishing. ISBN 978-3-319-06569-4.

[162] Claudia Leacock and Martin Chodorow. C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4):389–405, 2003. ISSN 00104817. URL `http://www.jstor.org/stable/30204913`.

[163] José Paulo Leal, Ricardo Queirós, and Duarte Ferreira. Specifying a Programming Exercises Evaluation Service on the e-Framework. In *Advances in Web-Based Learning – ICWL 2010*, pages 141–150, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[164] Petra Lietz, John C. Cresswell, Keith F. Rust, and Raymond J. Adams, editors. *Implementation of Large-Scale Education Assessments*. John Wiley & Sons, 2017.

[165] Ben Liu. SARAC: A Framework for Automatic Item Generation. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies*, pages 556–558, 2009.

[166] Georgine Loacker, Lucy Cromwell, and Kathleen O'Brian. Assessment in higher education: to serve the learner. In *National Conference on Assessment in Higher Education*, 1985.

[167] Wei-Yin Loh. Fifty Years of Classification and Regression Trees. *International Statistical Review*, 82(3):329–348, 2014. doi: 10.1111/insr.12016.

[168] LPLUS GmbH. LPLUS: Portfolio. URL `https://lplus.de/en/lplus-portfolio/`. Last accessed: 2017-12-01.

[169] Ronghua Lu, Haiying Liu, and Bingxiang Liu. Research and implementation of general online examination system. *Advanced Materials Research*, 926-930: 2374–2377, 2014. doi: 10.4028/www.scientific.net/AMR.926-930.2374.

[170] Yu-Chun Lu, Yu-Sheng Yang, Ping-Chun Chang, and Chu-Sing Yang. The design and implementation of intelligent assessment management system. In *IEEE Global Engineering Education Conference, EDUCON 2013, Berlin, Germany, March 13-15, 2013*, pages 451–457, 2013. doi: 10.1109/EduCon.2013.6530144.

[171] A. Pastor López-Monroy, Hugo Jair Escalante, Manuel Montes y Gómez, and Xavier Baró. Forensic Analysis Recognition. In *Engineering Data-Driven Adaptive Trust-based e-Assessment Systems*, number 34 in Lecture Notes on Data Engineering and Communications Technologies, pages 1–18. Springer, 2020. doi: 10.1007/978-3-030-29326-0_1.

[172] Xiaofeng Ma and Zhurong Zhou. Student pass rates prediction using optimized support vector machine and decision tree. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 209–215. IEEE, 2018.

[173] Don Mackenzie. Production and delivery of TRIADS Assessments on a university-wide basis. In *Proceedings of the 4th Computer-Aissisted Assessment Conference (CAA)*, 2000.

[174] Mathias Magdowski. Personalisierbare Aufgaben und anonymer Peer-Review. In Marlene Miglbauer, Lene Kieberl, and Stefan Schmid, editors, *Hochschule*

digital.innovativ | #digiPH Tagungsband zur 1. Online-Tagung, pages 327–340, Books on Demand GmbH, Norderstedt, December 2018. ISBN 9783748120056.

[175] Wassim Mahfouz and Heinz-Dietrich Wuttke. Automatic Classifiers for Formative Assessment. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2019.

[176] Uwe Maier, Carolin Ramsteck, and Kathrin Hoffmann. Formative Leistungsdiagnostik und Learning Analytics: Entwicklung, Nutzung und Optimierung eines onlinebasierten Kurses für die Diagnostik und Förderung von Grundwissen im Kompetenzbereich Sprachbetrachtung. *Zeitschrift für Erziehungswissenschaft*, 20 (4):728–747, 2017.

[177] David J. Malan. CS5 Sandbox: Secure Execution of Untrusted Code. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13)*, 2013.

[178] Kaleem Razzaq Malik and Tauqir Ahmad. E-Assessment Data Compatibility Resolution Methodology with Bidirectional Data Transformation. *EURASIA Journal of Mathematics, Science and Technology Education*, 13(7):3969–3991, 2017.

[179] K.R. Malik and T. Ahmad. E-assessment data compatibility resolution methodology with bidirectional data transformation. *Eurasia Journal of Mathematics, Science and Technology Education*, 13(7):3969–3991, 2017. doi: 10.12973/eurasia.2017.00767a.

[180] Sven Manske and Heinz Ulrich Hoppe. Automated Indicators to Assess the Creativity of Solutions to Programming Exercises. In *IEEE 14th International Conference on Advanced Learning Technologies (ICALT)*, pages 497–501, July 2014. doi: 10.1109/ICALT.2014.147.

[181] MapleSoft. Features in Maple T.A. URL `https://www.maplesoft.com/products/mapleta/mainfeatures.aspx`. Last accessed: 2017-12-01.

[182] Martin Mares. Moe – Design of a Modular Grading System. *Olympiads in Informatics*, 3:60–66, 2009.

[183] Alke Martens. Time in the Adaptive Tutoring Process Model. In *Intelligent Tutoring Systems. ITS 2006*, volume 4053 of *Lecture Notes in Computer Science*, pages 134–143, 2006. doi: 10.1007/11774303_14.

[184] Brent Martin. Authoring Educational Games with Greenmind. In *Intelligent Tutoring Systems (ITS 2008)*, volume 5091 of *Lecture Notes in Computer Science*, pages 684–686, 2008.

[185] Till Massing, Natalie Reckmann, Benjamin Otto, Kim J. Hermann, Christoph Hanck, and Michael Goedicke. Klausurprognose mit Hilfe von E-Assessment-Nutzerdaten. In *DeLFI 2018 - Die 16. E-Learning Fachtagung Informatik der*

Gesellschaft für Informatik e. V., volume 284 of *Lecture Notes in Informatics*, pages 171–176, 2018.

[186] Till Massing, Nils Schwinning, Michael Striewe, Christoph Hanck, and Michael Goedicke. E-Assessment Using Variable-Content Exercises in Mathematical Statistics. *Journal of Statistics Education*, 26(3):174–189, 2018. doi: 10.1080/10691898.2018.1518121.

[187] Maths for More. WIRIS Quizzes - Technical description. URL `http://www.wiris.com/en/quizzes/docs`. Last accessed: 2017-12-01.

[188] Erica Melis, Jeff Haywood, and Tim J. Smith. Leactivemath. In Wolfgang Nejdl and Klaus Tochtermann, editors, *Innovative Approaches for Learning and Knowledge Sharing: First European Conference on Technology Enhanced Learning (EC-TEL 2006)*, pages 660–666, Be, 2006. Springer Berlin Heidelberg. doi: 10.1007/11876663_69.

[189] L. Michel, L. Görtz, S. Radomski, T. Fritsch, and L. Baschour. Digitales Prüfen und Bewerten im Hochschulbereich. CHE Zentrum für Hochschulentwicklung, 2015.

[190] David E. Millard, Christopher Bailey, Hugh C. Davis, Lester Gilbert, Yvonne Howard, and Gary Wills. The e-Learning Assessment Landscape. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pages 964–966, July 2006. doi: 10.1109/ICALT.2006.1652604.

[191] C. Miller, L. Lecheler, B. Hosack, A. Doering, and S. Hooper. Orchestrating data, design, and narrative: Information visualization for sense- and decision-making in online learning. *International Journal of Cyber Behavior, Psychology and Learning*, 2(2):1–15, 2012. doi: 10.4018/ijcbpl.2012040101.

[192] Antonija Mitrovic, Brent Martin, and Pramuditha Suraweera. Intelligent Tutors for All: The Constraint-Based Approach. *IEEE Intelligent Systems*, 22(4):38–45, jul 2007. doi: 10.1109/mis.2007.74.

[193] Laurent Moccozet, Omar Benkacem, and Pierre-Yves Burgi. Towards a Technology-Enhanced Assessment Service in Higher Education. In Michael E. Auer, David Guralnick, and James Uhomoibhi, editors, *Interactive Collaborative Learning*, pages 453–467, Cham, 2017. Springer International Publishing. doi: 10.1007/978-3-319-50340-0_40.

[194] Néstor Mora, Santi Caballe, and Thanasis Daradoumis. Providing a Multi-fold Assessment Framework to Virtualized Collaborative Learning in Support for Engineering Education. *International Journal of Emerging Technologies in Learning (iJET)*, 11(07):41–51, 2016. ISSN 1863-0383. doi: 10.3991/ijet.v11i07.5882. URL `https://online-journals.org/index.php/i-jet/article/view/5882`.

[195] Edna H. Mory. Adaptive Feedback in Computer-Based Instruction: Effects of Response Certitude on Performance, Feedback-Study Time, and Efficiency. *Journal of Educational Computing Research*, 11(3):263–290, 1994. doi: 10.2190/YM7U-G8UN-8U5H-HD8N.

[196] E. Muravyeva, J. Janssen, K. Dirkx, and M. Specht. Students' attitudes towards personal data sharing in the context of e-assessment: informed consent or privacy paradox? In *Communications in Computer and Information Science*, volume 1014, pages 16–26, 2019. doi: 10.1007/978-3-030-25264-9_2.

[197] Ekaterina Muravyeva, José Janssen, Marcus Specht, and Bart Custers. Exploring solutions to the privacy paradox in the context of e-assessment: informed consent revisited. *Ethics and Information Technology*, 2000. doi: 10.1007/s10676-020-09531-5.

[198] Tom Murray. Having it all, maybe: Design tradeoffs in ITS authoring tools. In *Intelligent Tutoring Systems (ITS 1996)*, 1996.

[199] Arsenio Muñoz de la Peña, David González-Gómez, David Muñoz de la Peña, Fabio Gómez-Estern, and Manuel Sánchez Sequedo. Automatic Web-Based Grading System: Application in an Advanced Instrumental Analysis Chemistry Laboratory. *Journal of Chemical Education*, 90(3):308–314, 2013. doi: 10.1021/ed3000815.

[200] G.S. Nandakumar, V. Geetha, B. Surendiran, and S. Thangasamy. A rough set based classification model for grading in adaptive e-assessment. *International Review on Computers and Software*, 9(7):1169–1177, 2014.

[201] Mahmoud Neji and Mohamed Ben Ammar. Agent-based Collaborative Affective e-Learning Framework. *Electronic Journal of e-Learning*, 5(2):123–134, 2007.

[202] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015. ISBN 978-1491950357.

[203] Kia Ng and Paolo Nesi. i-Maestro Framework and Interactive Multimedia Tools for Technology-Enhanced Learning and Teaching for Music. In *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 266–269, 2008.

[204] Melvin R. Novick. The axioms and principal results of classical test theory. *Journal of Mathematical Psychology*, 3(1):1–18, 1966. ISSN 0022-2496. doi: 10.1016/0022-2496(66)90002-2.

[205] A. Núñez, J. Fernández, J. D. Garcia, L. Prada, and J. Carretero. M-PLAT: Multi-Programming Language Adaptive Tutor. In *Eighth IEEE International Conference on Advanced Learning Technologies*, pages 649–651, July 2008. doi: 10.1109/ICALT.2008.153.

[206] Object Management Group, Inc. XML Metadata Interchange (XMI), v2.5.1 specification, 2015. `http://www.omg.org/spec/XMI/2.5.1/`.

[207] A. Okada, D. Whitelock, W. Holmes, and C. Edwards. e-authentication for online assessment: A mixed-method study. *British Journal of Educational Technology*, 50 (2):861–875, 2019. doi: 10.1111/bjet.12608.

[208] Abdulsalam Sulaiman Olaniyi, Saheed Yakub Kayode, Hambali Moshood Abiola, Salau-Ibrahim Taofeekat Tosin, and Akinbowale Nathaniel Babatunde. Student's performance analysis using decision tree algorithms. *Annals. Computer Science Series*, 15(1), 2017.

[209] J. Opgen-Rhein, B. Küppers, and U. Schroeder. Requirements for author verification in electronic computer science exams. In *11th International Conference on Computer Supported Education, CSEDU 2019*, volume 2, pages 432–439, 2019. doi: 10.5220/0007736104320439.

[210] Alexandros Papadimitriou, Maria Grigoriadou, and Georgios Gyftodimos. Interactive Problem Solving Support in the Adaptive Educational Hypermedia System MATHEMA. *TLT*, 2(2):93–106, 2009. doi: 10.1109/TLT.2009.19.

[211] Zacharoula Papamitsiou and Anastasios Economides. An Assessment Analytics Framework (AAF) for Enhancing Students' Progress. In Santi Caballé and Robert Clarisó, editors, *Formative Assessment, Learning Data Analytics and Gamification*, Intelligent Data-Centric Systems, pages 117–133. Academic Press, Boston, 2016. ISBN 978-0-12-803637-2. doi: 10.1016/B978-0-12-803637-2.00007-5.

[212] Abelardo Pardo. A Multi-agent Platform for Automatic Assignment Management. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '02, pages 60–64. ACM, 2002. ISBN 1-58113-499-1. doi: 10.1145/544414.544434.

[213] Cristina Pérez-Solà, Jordi Herrera-Joancomartí, and Helena Rifà-Pous. On Improving Automated Self-assessment with Moodle Quizzes: Experiences from a Cryptography Course. In Eric Ras and Ana Elena Guerrero Roldán, editors, *Technology Enhanced Assessment*, pages 176–189, Cham, 2018. Springer International Publishing. ISBN 978-3-319-97807-9.

[214] Christian Perfect. A demonstration of Numbas, an e-assessment system for mathematical disciplines. In *CAA Conference*, 2015.

[215] Simon Perry, Igor Bulatov, and Edward Roberts. The use of e-assessment in chemical engineering education. *Chemical Engineering Transactions*, 12:555–560, 2007.

[216] Sebastian Pobel and Michael Striewe. Domain-Specific Extensions for an E-Assessment System. In Michael A. Herzog, Zuzana Kubincová, Peng Han, and

Marco Temperini, editors, *Advances in Web-Based Learning – ICWL 2019*, pages 327–331, Cham, 2019. Springer International Publishing. ISBN 978-3-030-35758-0. doi: 10.1007/978-3-030-35758-0_32.

[217] René Ponto, Tobias Schüler, and Michael Striewe. Ansätze zur automatischen Generierung von Aufgaben zum Modellverstehen am Beispiel von UML-Sequenzdiagrammen (Approaches for Automatic Generation of Tasks for Model Comprehension using UML Sequence Diagrams as an Example). In *Companion Proceedings of Modellierung 2020 Short, Workshop and Tools & Demo Papers co-located with Modellierung 2020, Vienna, Austria, February 19-21, 2020*, pages 77–88, 2020. URL `http://ceur-ws.org/Vol-2542/MOHOL4.pdf`.

[218] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeno, and Paolo Rosso. Overview of the 1st International Competition on Plagiarism Detection. In *3rd PAN Workshop. Uncovering Plagiarism, Authorship and Social Software Misuse*, pages 1–9, 2009.

[219] Martin Potthast, Alberto Barrón-Cedeno, Benno Stein, and Paolo Rosso. Cross-language plagiarism detection. *Language Resources & Evaluation*, pages 1—18, 2010.

[220] Debra Pugh, André De Champlain, Mark Gierl, Hollis Lai, and Claire Touchie. Can automated item generation be used to develop high quality MCQs that assess application of knowledge? *Research and Practice in Technology Enhanced Learning*, 15(1):12, 2020. ISSN 1793-7078. doi: 10.1186/s41039-020-00134-8.

[221] Khirulnizam Abd Rahman, Syarbaini Ahmad, and Md Jan Nordin. The Design of an Automated C Programming Assessment Using Pseudo-code Comparison Technique. In *National Conference on Software Engineering and Computer Systems*, 2007.

[222] M. Ramamurthy, I. Krishnamurthi, and S. Mammen. Great evaluator: An automated assessment system for evaluating regular grammars in automata theory. *Global Journal of Pure and Applied Mathematics*, 12(3):2085–2111, 2016.

[223] G. Rasch. *Probabilistic models for some intelligence and attainment tests.* University of Chicago Press, Chicago, 1980. ISBN 0-226-70553-6.

[224] Miroslava Raspopović, Svetlana Cvetanović, Dušan Stanojević, and Mateja Opačić. Software architecture for integration of institutional and social learning environments. *Science of Computer Programming*, 129:92 – 102, 2016. ISSN 0167-6423. doi: 10.1016/j.scico.2016.07.001. URL `http://www.sciencedirect.com/science/article/pii/S0167642316300855`. Special issue on eLearning Software Architectures.

[225] Tobias Reischmann and Herbert Kuchen. Towards an e-assessment tool for advanced software engineering skills. In *16th Koli Calling International Conference on Computing Education Research*, pages 81–90. Association for Computing Machinery, 2016. ISBN 9781450347709. doi: 10.1145/2999541.2999550.

[226] Robert A. Reiser and John V. Dempsey. *Trends and Issues in Instructional Design and Technology*. ALLYN & BACON, 2011. ISBN 978-0132563581. URL `http://www.ebook.de/de/product/14065487/robert_a_reiser_john_v_dempsey_trends_and_issues_in_instructional_design_and_technology.html`.

[227] C.R. Reynolds, R.B. Livingston, and V.L. Willson. *Measurement and Assessment in Education*. Alternative eText Formats Series. Pearson, 2009. ISBN 9780205579341. URL `https://books.google.de/books?id=Lh-MbwAACAAJ`.

[228] Thomas Richter and David Boehringer. Towards electronic exams in undergraduate engineering. In *2014 IEEE Global Engineering Education Conference, EDUCON 2014, Istanbul, Turkey, April 3-5, 2014*, pages 196–201, 2014. doi: 10.1109/EDUCON.2014.6826090.

[229] Jeff W. Rickel. Intelligent Computer-Aided Instruction: A Survey Organized Around System Components. *IEEE Transactions on Systems, Man, and Cybernetics*, 19 (1):40–57, 1989. doi: 10.1109/21.24530.

[230] SB Rinderle, Ralph Bobrik, Manfred Reichert, and Thomas Bauer. Business process visualization – use cases, challenges, solutions. In *Proceedings of the Eighth International Conference on Enterprise Information Systems: Information System Analysis and Specification*, ICEIS, pages 204–211. INSTICC Press, 2006. URL `http://doc.utwente.nl/66217/`.

[231] Sasko Ristov, Marjan Gusev, Goce Armenski, and Goran Velkoski. Scalable and Elastic e-Assessment Cloud Solution. In *IEEE Global Engineering Education Conference (EDUCON)*, 2014.

[232] J. Robison, S. McQuiggan, and J. Lester. Evaluating the consequences of affective feedback in intelligent tutoring systems. In *3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6, Sept 2009. doi: 10.1109/ACII.2009.5349555.

[233] Cristóbal Romero and Sebastián Ventura. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):601–618, 2010. doi: 10.1109/tsmcc.2010.2053532.

[234] J. Rost and C. H. Carstensen. Multidimensional Rasch Measurement via Item Component Models and Faceted Designs. *Applied Psychological Measurement*, 26 (1):42–56, 2002.

[235] Vicki Roth, Volodymyr Ivanchenko, and Nicholas Record. Evaluating Student Response to WeBWorK, a Web-based Homework Delivery and Grading System. *Comput. Educ.*, 50(4):1462–1482, 2008. ISSN 0360-1315. doi: 10.1016/j.compedu.2007.01.005.

[236] Peter James Rowlett. *A partially-automated approach to the assessment of mathematics in higher education*. PhD thesis, Nottingham Trent University, 2013. URL `http://irep.ntu.ac.uk/id/eprint/309/`.

[237] Lawrence M. Rudner and Tahung Liang. Automated Essay Scoring Using Bayes' Theorem. *The Journal of Technology, Learning, and Assessment*, 1(2), June 2002. URL `https://ejournals.bc.edu/index.php/jtla/article/view/1668`.

[238] Lawrence M. Rudner, Veronica Garcia, and Catherine Welch. An Evaluation of IntelliMetric™ Essay Scoring System. *The Journal of Technology, Learning and Assessment*, 4(4), 2006. URL `https://ejournals.bc.edu/index.php/jtla/article/view/1651`.

[239] S. Runzrat, A. Harfield, and S. Charoensiriwath. Applying item response theory in adaptive tutoring systems for thai language learners. In *Proc. 11th International Conference on Knowledge and Smart Technology, KST 2019*, pages 67–71, 2019. doi: 10.1109/KST.2019.8687462.

[240] M. Sainsbury and T. Benton. Designing a formative e-assessment: Latent class analysis of early reading skills. *British Journal of Educational Technology*, 42(3): 500–514, 2011. doi: 10.1111/j.1467-8535.2009.01044.x.

[241] Chris Sangwin. *Computer Aided Assessment of Mathematics*. Oxford University Press, 2013.

[242] S. Sarre and M. Foulonneau. Reusability in e-assessment: Towards a multifaceted approach for managing metadata of e-assessment resources. In *5th International Conference on Internet and Web Applications and Services, ICIW 2010*, pages 420–425, 2010. doi: 10.1109/ICIW.2010.70.

[243] C. Saul and H.-D. Wuttke. Turning learners into effective better learners: The use of the askme! system for learning analytics. In *UMAP 2014 Posters, Demonstrations and Late-breaking Results*, volume 1181 of *CEUR Workshop Proceedings*, pages 57–60, 2014.

[244] Christian Saul and Heinz-Dietrich Wuttke. E-assessment meets personalization. In *IEEE Global Engineering Education Conference, EDUCON 2013, Berlin, Germany, March 13-15, 2013*, pages 200–206, 2013. doi: 10.1109/EduCon.2013.6530106.

[245] Christian Saul and Heinz-Dietrich Wuttke. Turning Learners into Effective Better Learners: The Use of the askMe! System for Learning Analytics. In *UMAP Workshops*, 2014.

[246] Silvia Schiaffino, Patricio Garcia, and Analia Amandi. eTeacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4):1744–1754, 2008.

[247] Louis A Schultheiss and Edward M Heiliger. Techniques of flow-charting. *Clinic on Library Applications of Data Processing*, 1963.

[248] Nils Schwinning, Melanie Schypula, Michael Striewe, and Michael Goedicke. Concepts and Realisations of Flexible Exercise Design and Feedback Generation in an e-Assessment System for Mathematics. In *Joint Proceedings of the MathUI, OpenMath and ThEdu Workshops and Work in Progress track at CICM, co-located with Conferences on Intelligent Computer Mathematics (CICM 2014)*, 2014.

[249] Niall Sclater and Karen Howie. User Requirements of the "Ultimate" Online Assessment Engine. *Comput. Educ.*, 40(3):285–306, April 2003. ISSN 0360-1315. doi: 10.1016/S0360-1315(02)00132-X. URL `http://dx.doi.org/10.1016/S0360-1315(02)00132-X`.

[250] Stylianos Sergis and Demetrios G Sampson. Teaching and learning analytics to support teacher inquiry: A systematic literature review. In *Learning analytics: Fundaments, applications, and trends*, pages 25–63. Springer, 2017.

[251] Varun Shenoy, Ullas Aparanji, K. Sripradha, and Viraj Kumar. Generating DFA Construction Problems Automatically. In *International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 32–37, 2016. doi: 10.1109/LaTiCE.2016.8.

[252] Wei Shi, Kosuke Kaneko, Chenguang Ma, and Yoshihiro Okada. A Framework for Automatically Generating Medical Quizzes with Multi-media Contents Based on Linked Data. In Leonard Barolli, Fang-Yie Leu, Tomoya Enokido, and Hsing-Chung Chen, editors, *Advances on Broadband and Wireless Computing, Communication and Applications*, pages 147–158, Cham, 2019. Springer International Publishing. ISBN 978-3-030-02613-4. doi: 10.1007/978-3-030-02613-4_13.

[253] Raheel Siddiqi, Christopher J. Harrison, and Rosheena Siddiqi. Improving Teaching and Learning through Automated Short-Answer Marking. *TLT*, 3(3):237–249, 2010. doi: 10.1109/TLT.2010.4.

[254] George Siemens. Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10):1380–1400, 2013.

[255] Guttorm Sindre and Aparna Vegendla. E-exams and exam process improvement. In *Proceedings of the UDIT / NIK 2015 conference*, 2015.

[256] Frano Skopljanac-Macina, Bruno Blaskovic, and Damir Pintar. Automated Generation of Questions for Basic Electrical Engineering Education. In *Proceedings of the 27th DAAAM International Symposium on Intelligent Manufacturing and Automation*, 2016. doi: 10.2507/27th.daaam.proceedings.056.

[257] B. J. Skromme, P. J. Rayes, B. E. McNamara, V. Seetharam, X. Gao, T. Thompson, X. Wang, B. Cheng, Y.-F. Huang, and D. H. Robinson. Step-based tutoring system

for introductory linear circuit analysis. In *Proceedings - Frontiers in Education Conference, FIE*, 2015. ISBN 9781479984534. doi: 10.1109/FIE.2015.7344312.

[258] B. J. Skromme, V. Seetharam, X. Gao, B. Korrapati, B. E. McNamara, Y.-F. Huang, and D. H. Robinson. Impact of step-based tutoring on student learning in linear circuit courses. In *46th Annual Frontiers in Education Conference (FIE)*, 2016. ISBN 9781509017904. doi: 10.1109/FIE.2016.7757638.

[259] Sharon Slade and Paul Prinsloo. Learning analytics: Ethical issues and dilemmas. *American Behavioral Scientist*, 57(10):1510–1529, 2013.

[260] Neil Smith, Pete Thomas, and Kevin Waugh. Automatic Grading of Free-Form Diagrams with Label Hypernymy. In *Learning and Teaching in Computing and Engineering (LaTiCE 2013)*, pages 136–142. IEEE, 2013. doi: 10.1109/LaTiCE.2013.33.

[261] Robert A. Sottilare and Michael Proctor. Passively Classifying Student Mood and Performance within Intelligent Tutors. *Journal of Educational Technology & Society*, 15(2):101–114, 2012. ISSN 11763647, 14364522. URL `http://www.jstor.org/stable/jeductechsoci.15.2.101`.

[262] Rúben Sousa and José Paulo Leal. A structural approach to assess graph-based exercises. In *International Symposium on Languages, Applications and Technologies*, pages 182–193. Springer, 2015. doi: 10.1007/978-3-319-27653-3_18.

[263] A. Stack, B. Boitshwarelo, A. Reedy, T. Billany, H. Reedy, R. Sharma, and J. Vemuri. Investigating online tests practices of university staff using data from a learning management system: The case of a business school. *Australasian Journal of Educational Technology*, 36(4), 2020. doi: 10.14742/AJET.4975.

[264] Patrick Stalljohann. Verwaltung universitärer Assessment-Szenarien am Beispiel von Informatik-Vorlesungen. In *DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V., 12.-15. September 2010, Universität Duisburg-Essen*, pages 109–120, 2010. URL `http://subs.emis.de/LNI/Proceedings/Proceedings169/article5734.html`.

[265] Slavomir Stankov, Marko Rosić, Branko Žitko, and Ani Grubišić. TEx-Sys model for building intelligent tutoring systems. *Computers & Education*, 51(3):1017–1036, 2008. ISSN 0360-1315. doi: https://doi.org/10.1016/j.compedu.2007.10.002. URL `http://www.sciencedirect.com/science/article/pii/S0360131507001297`.

[266] Benno Stein, Nedim Lipka, and Peter Prettenhofer. Intrinsic plagiarism analysis. *Language Resources and Evaluation*, 45(1):63–82, 2011.

[267] Michael Striewe. Generierung von Zusatzinformationen in automatischen Systemen zur Bewertung von Programmieraufgaben. In *Proceedings of the First Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2013)*, 2013. URL `http://ceur-ws.org/Vol-1067/abp2013_submission_6.pdf`.

[268] Michael Striewe. *Automated Assessment of Software Artefacts - A Use Case in E-Assessment*. PhD thesis, University of Duisburg-Essen, 2014. URL `https://nbn-resolving.org/urn:nbn:de:hbz:464-20150313-120503-7`.

[269] Michael Striewe. An architecture for modular grading and feedback generation for complex exercises. *Science of Computer Programming*, 129:35–47, 2016. ISSN 0167-6423. doi: 10.1016/j.scico.2016.02.009. URL `http://www.sciencedirect.com/science/article/pii/S0167642316300260`.

[270] Michael Striewe. Dynamic Generation of Assessment Items Using Wikidata. In *Technology Enhanced Assessment - 21st International Conference, TEA 2018, Amsterdam, The Netherlands, December 10-11, 2018, Revised Selected Papers*, pages 1–15, 2018. doi: 10.1007/978-3-030-25264-9_1.

[271] Michael Striewe. Towards a Pattern Catalogue for E-Assessment System Integration. In *Combined Proceedings of the Workshops of the German Software Engineering Conference 2018 (SE 2018), Ulm, Germany, March 06, 2018.*, pages 62–65, 2018. URL `http://ceur-ws.org/Vol-2066/seels2018paper03.pdf`.

[272] Michael Striewe. Lean and Agile Assessment Workflows. In David Parsons and Kathryn MacCallum, editors, *Agile and Lean Concepts for Teaching and Learning: Bringing Methodologies from Industry to the Classroom*, pages 187–204. Springer Singapore, Singapore, 2019. ISBN 978-981-13-2751-3. doi: 10.1007/978-981-13-2751-3_10.

[273] Michael Striewe. Automatische Aufgabengenerierung über Linked Open Data am Beispiel der Archäologie. In Niels Pinkwart and Johannes Konert, editors, *DELFI 2019 - Die 17. Fachtagung Bildungstechnologien*, pages 115–120, Bonn, 2019. Gesellschaft für Informatik e.V. doi: 10.18420/delfi2019_104.

[274] Michael Striewe. Components and Design Alternatives in E-Assessment Systems. In *Software Architecture - 13th European Conference, ECSA 2019, Paris, France, September 9-13, 2019, Proceedings*, pages 220–228, 2019. doi: 10.1007/978-3-030-29983-5_15.

[275] Michael Striewe. Design Patterns for Submission Evaluation within E-Assessment Systems. In *26th European Conference on Pattern Languages of Programs*, EuroPLoP'21, pages 32:1–32:10, 2021. ISBN 9781450389976. doi: 10.1145/3489449.3490010.

[276] Michael Striewe. A Lightweight Method for Modelling Technology-Enhanced Assessment Processes. In *14th International Conference on Computer Supported Education (CSEDU)*, 2022.

[277] Michael Striewe. Where does all the data go? – A Review of Research on E-Assessment Data. In *14th International Conference on Computer Supported Education (CSEDU)*, 2022.

[278] Michael Striewe and Michael Goedicke. Analyse von Programmieraufgaben durch Softwareproduktmetriken. In *SEUH*, pages 59–68, 2013.

[279] Michael Striewe and Michael Goedicke. Trace Alignment for Automated Tutoring. In *Proceedings of International Computer Assisted Assessment (CAA) Conference 2013*, Southampton, 2013.

[280] Michael Striewe and Michael Goedicke. Automated Assessment of UML Activity Diagrams. In *Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE 2014)*, page 336, 2014. doi: 10.1145/2591708.2602657.

[281] Michael Striewe and Michael Goedicke. A Review of Static Analysis Approaches for Programming Exercises. In *Proceedings of the International Conference on Computer Assisted Assessment (CAA 2014)*, number 439 in CCIS, pages 100–113, Zeist, Netherlands, 2014.

[282] Michael Striewe and Michael Goedicke. Automatische Generierung von Aufgaben zum Codeverständnis. In *DeLFI 2018 - Die 16. E-Learning Fachtagung Informatik der Gesellschaft für Informatik e. V.*, volume 284 of *Lecture Notes in Informatics*, pages 153–164, 2018.

[283] Michael Striewe, Florian Trauten, and Carolin Eitemüller. Aufgaben mit automatischem Feedback zu chemischen Atom-Orbitalschemata. In Raphael Zender, Dirk Ifenthaler, Thiemo Leonhardt, and Clara Schumacher, editors, *DELFI 2020 – Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V.*, pages 109–119, Bonn, 2020. Gesellschaft für Informatik e.V.

[284] Michael Striewe, Martin Forell, Constantin Houy, Peter Pfeiffer, Gunther Schiefer, Selina Schüler, Chantal Soyka, Tobias Stottrop, Meike Ullrich, Peter Fettke, Peter Loos, Andreas Oberweis, and Niclas Schaper. Kompetenzorientiertes E-Assessment für die grafische, konzeptuelle Modellierung. *HMD Praxis der Wirtschaftsinformatik*, 58(6), 2021. ISSN 2198-2775. doi: 10.1365/s40702-021-00797-x.

[285] Wei Su, Paul Wang, and Lian Li. MathPASS: A Remedial Mathematics System with Automated Answer Checking, 2010.

[286] Yuni Susanti, Ryu Iida, and Takenobu Tokunaga. Automatic Generation of English Vocabulary Tests. In *7th International Conference on Computer Supported Education (CSEDU)*, 2015.

[287] Rubén Sánchez-Dams, Alexander Barón-Salazar, and Maria Clara Gómez-Álvarez. An Extension of the SEMAT Kernel for Representing Teaching and Learning Practices about Embedded Systems. In *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 39–46, April 2016. doi: 10.1109/CONISOFT.2016.15.

[288] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009. ISBN 0470167742, 9780470167748.

[289] Dirk T. Tempelaar, Bart Rienties, and Bas Giesbers. Stability and sensitivity of Learning Analytics based prediction models. In *Proceedings of 7th International conference on Computer Supported Education (CSEDU)*, pages 156–166, 2015.

[290] Tom Thaler, Constantin Houy, Peter Fettke, and Peter Loos. Automated Assessment of Process Modeling Exams: Basic Ideas and Prototypical Implementation. In Stefanie Betz and Ulrich Reimer, editors, *Modellierung 2016 Workshopband*, LNI, pages 63–70, 2016.

[291] Pete Thomas, Neil Smith, and Kevin Waugh. Automatically assessing graph-based diagrams. *Learning, Media and Technology*, 33(3):249–267, 2008.

[292] Pete Thomas, Kevin Waugh, and Neil Smith. A revision tool for teaching and learning sequence diagrams. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008*, 2008. URL `http://oro.open.ac.uk/19241/`.

[293] Justin Timm, Benjamin Otto, Thilo Schramm, Michael Striewe, Philipp Schmiemann, and Michael Goedicke. Technical Aspects of Automated Item Generation for Blended Learning Environments in Biology. *i-com*, 1(19):3–15, April 2020. ISSN 1618-162X. doi: 10.1515/icom-2020-0001.

[294] Sue Timmis, Patricia Broadfoot, Rosamund Sutherland, and Alison Oldfield. Rethinking assessment in a digital age: opportunities, challenges and risks. *British Educational Research Journal*, 42(3):454–476, 2016. doi: 10.1002/berj.3215.

[295] Ana Paula Tomás and José Paulo Leal. Automatic Generation and Delivery of Multiple-Choice Math Quizzes. In Christian Schulte, editor, *Principles and Practice of Constraint Programming*, pages 848–863, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40627-0.

[296] G. Tremblay, F. Guérin, A. Pons, and A. Salah. Oto, a generic and extensible tool for marking programming assignments. *Software: Practice and Experience*, 38(3): 307–333, 2008. doi: 10.1002/spe.839.

[297] M. Usman, M.M. Iqbal, Z. Iqbal, M.U. Chaudhry, M. Farhan, and M. Ashraf. E-assessment and computer-aided prediction methodology for student admission test score. *Eurasia Journal of Mathematics, Science and Technology Education*, 13 (8):5499–5517, 2017. doi: 10.12973/eurasia.2017.00939a.

[298] Salvatore Valenti, Francesca Neri, and Alessandro Cucchiarelli. An Overview of Current Research on Automated Essay Grading. *Journal of Information Technology Education: Research*, 2(1):319–330, January 2003. ISSN 1539-3585. URL `https://www.learntechlib.org/p/111481`.

[299] Stephanie Valentine, Francisco Vides, George Lucchese, David Turner, Hong hoe Kim, Wenzhe Li, Julie Linsey, and Tracy Hammond. Mechanix: A Sketch-Based Tutoring and Grading System for Free-Body Diagrams. *AI Magazine*, pages 55–66, 2013.

[300] W.M.P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639 – 650, 1999. ISSN 0950-5849. doi: 10.1016/S0950-5849(99)00016-6.

[301] Wim J. Van der Linden and Cees A. W . Glas. *Computerized Adaptive Testing: Theory and Practice*. Springer, 2000.

[302] Mario Manso Vázquez and Martín Llamas Nistal. Distributed Personal Learning Environments Towards a suitable architecture. In *IEEE Global Engineering Education Conference, EDUCON 2013, Berlin, Germany, March 13-15, 2013*, pages 664–673, 2013. doi: 10.1109/EduCon.2013.6530178.

[303] C Wallace, P Dargan, and A Jones. Paracetamol overdose: an evidence based flowchart to guide management. *Emergency Medicine Journal: EMJ*, 19(3):202, 2002.

[304] Emily H. Watts, Mary O'Brian, and Brian W. Wojcik. Four Models of Assistive Technology Consideration: How Do They Compare to Recommended Educational Assessment Practices? *Journal of Special Education Technology*, 19(1):43–56, 2003. doi: 10.1177/016264340401900104.

[305] Franz E. Weinert. Concept of competence: A conceptual clarification. In *Defining and Selecting Key Competencies*, pages 45–65. Hogrefe & Huber Publishers, 2001.

[306] Martin M Weng, Ireti Fakinlede, Fuhua Lin, Timothy K Shih, and Maiga Chang. A conceptual design of multi-agent based personalized quiz game. In *11th IEEE International Conference on Advanced Learning Technologies (ICALT 2011)*, pages 19–21. IEEE, 2011.

[307] Denise Whitelock, Alison Twiner, John T. E. Richardson, Debora Field, and Stephen Pulman. What Does a 'Good' Essay Look Like? Rainbow Diagrams Representing Essay Quality. In Eric Ras and Ana Elena Guerrero Roldán, editors, *Technology Enhanced Assessment*, pages 1–12, Cham, 2018. Springer International Publishing. ISBN 978-3-319-97807-9.

[308] Chris Wilcox. Testing Strategies for the Automated Grading of Student Programs. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 437–442, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3685-7. doi: 10.1145/2839509.2844616.

[309] Gary B. Wills, Christopher P. Bailey, Hugh C. Davis, Lester Gilbert, Yvonne Howard, Steve Jeyes, David E. Millard, Joseph Price, Niall Sclater, Robert Sherratt, Iain Tulloch, and Rowin Young. An e-Learning Framework for Assessment

(FREMA). In *Proceedings of the 11th Computer-Assisted Assessment Conference (CAA)*, 2007.

[310] Volker Wölfert. Technische Unterstützung zur Durchführung von Massenklausuren. mathesis, Universität Potsdam, 2015.

[311] Wanli Xing, Rui Guo, Eva Petakovic, and Sean Goggins. Participation-based student final performance prediction model through interpretable Genetic Programming: Integrating learning analytics, educational data mining and theory. *Computers in Human Behavior*, 47:168–181, 2015.

[312] Linting Xue. Intelligent argument grading system for student-produced argument diagrams. In X. Hu, X. Hu, T. Barnes, A. Hershkovitz, and L. Paquette, editors, *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017*, pages 439–441. International Educational Data Mining Society, 2017.

[313] Olaf Zawacki-Richter, Victoria I. Marín, Melissa Bond, and Franziska Gouverneur. Systematic review of research on artificial intelligence applications in higher education – where are the educators? *International Journal of Educational Technology in Higher Education*, 16(39), 2019. doi: 10.1186/s41239-019-0171-0.

[314] Yu Zhong and Yunbin Deng. A survey on keystroke dynamics biometrics: approaches, advances, and evaluations. In *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics*, pages 1–22. Science Gate Publishing, 2015.

[315] Steffen Zschaler, Sam White, Kyle Hodgetts, and Martin Chapman. Modularity for Automated Assessment: A Design-Space Exploration. In *Combined Proceedings of the Workshops of the German Software Engineering Conference 2018 (SE 2018), Ulm, Germany, March 06, 2018.*, Ulm, Germany, March 2018.

[316] Oliver Zscheyge and Karsten Weicker. Werkzeugunterstützung bei der Vermittlung der Grundlagen wissenschaftlichen Schreibens. In *Hochschuldidaktik der Informatik, HDI 2016 - 7. Fachtagung des GI-Fachbereichs Informatik und Ausbildung / Didaktik der Informatik, 13.-14. September 2016 an der Universität Potsdam, Germany*, pages 57–68, 2016.

[317] Imran A. Zualkernan and Maha Shouman. Towards Ontology-Driven Heuristic Assessment Generation for Software Design Patterns. In *Eighth IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 922–924, July 2008. doi: 10.1109/ICALT.2008.74.

[318] František Špaček, Radomír Sohlich, and Tomáš Dulík. Docker as Platform for Assignments Evaluation. *Procedia Engineering*, 100:1665 – 1671, 2015. ISSN 1877-7058. doi: 10.1016/j.proeng.2015.01.541. URL `http://www.sciencedirect.com/science/article/pii/S1877705815005688`.

DOI: 10.17185/duepublico/81826
URN: urn:nbn:de:hbz:465-20240412-110202-3